Heriot-Watt University

Doctoral Thesis

Surface Tension Dominated Flows using Smoothed Particle Hydrodynamics

Author:
Samat Maxutov

Supervisor:
Dr. Yeaw-Chu Lee &
Dr. Baixin Chen

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

in the

Institute of Mechanical, Process and Energy Engineering
School of Engineering and Physical Sciences

October 2017

"Even if everyone is doing it, wrong is never right. Evil, error and darkness will never be truth, even if popular."

Russel M. Nelson

"Life is like riding a bicycle. To keep balance, you must keep moving."

Albert Einstein

HERIOT-WATT UNIVERSITY

Abstract

Mechanical Engineering Department
School of Engineering and Physical Sciences

Doctor of Philosophy

Surface Tension Dominated Flows using Smoothed Particle Hydrodynamics

by Samat MAXUTOV

The tiny scale and time duration coupled with unfriendly environment for instrumentation represented by droplet and thin film make experimental monitoring challenging, and consequently the need for computational model for capturing the droplet and thin films processes. Traditional computational models such as the Finite Element Method (FEM) is not able to capture large deformations effectively, as well as accurate interface tracking and material histories remains challenging. The Smoothed Particle Hydrodynamics (SPH) which is a simple, effective and relatively new numerical model, can reliably capture and model the complex dynamic behaviour of surface tension dominated free-surface flows. The major advantage of SPH over the Eulerian approach is that the fluid, represented by particles, is naturally suitable for complex geometries and problems arising from free-surface flows. The present work advance the work of applying SPH to droplets with different surface tension models and with one of these models been applied for the first time in SPH. The solver was developed from scratch with C++ programming language for single phase problems to increase the computational performance. The developed SPH solver is described and verified against existing theoretical and experimental results. A detailed investigation of droplet simulation and contact angle hysteresis is analysed and described in order to demonstrate the strength of the developed SPH solver. Part of the results was submitted to International Journal for Numerical Methods in Fluids and received positive feedback.

Acknowledgements

I would like to thank my supervisor Dr Yeaw Chu Lee for his constant support and help throughout my PhD study and I am very proud to be one of your students. I would also like to thank members of our research group Nowoghomwenma Noel Ehigiamusoe, Odin Du'Plessis, Alasdair McKean and Luke Stephen Mason for their supervision, assistance and time.

A particular thank to my parents, my parents-in-law, my brother and my sister for their support in difficult times. A big hug to my lovely wife, Shakhrizat Agaidarova. It's you helped to overcome the difficulties of doing a PhD.

This work would not have been possible without the support and funding from the Center for International Programs under the Ministry of Education and Science of the Republic of Kazakhstan.

Contents

List of Figures

List of Tables

Abbreviations

BC	Boundary Condition
CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy
CLF	Contact Line Force
CPU	Central Processing Unit
CSF	Continuum Surface Force
FEM	Finite Element Method
FVM	Finite Volume Method
GPU	Graphics Processing Unit
HVI	High Velocity Impact
IC	Initial Condition
IIF	Inter-particle Interaction Force
ISPH	Incompressible Smoothed Particle Hydrodynamics
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
SPH	Smoothed Particle Hydrodynamics
WCSPH	Weakly Compressible Smoothed Particle Hydrodynamics
XSPH	eXtended Smoothed Particle Hydrodynamics

Symbols

p	pressure	Pa
V	volume	m^3
h	smoothing length	m
I	unit tensor	
m	particle mass	kg
N	number of particles	
$\mathbf{q}$	heat flux	J/m^2s
Re	Reynolds number	
t	time	s
T	temperature	K
e	internal energy	J/kg
$\mathbf{v}$	particle velocity	m/s
W	smoothing function	m^{-3}
$\mathbf{r}$	particle position	m

ω	angular frequency	$rads^{-1}$
ρ	particle density	kg/m^3
μ	viscosity	kg/m s
τ	viscous stress tensor	kg/m s^2

To my family

Chapter 1

Introduction

Surface tension plays an important role in multiphase and free-surface flows at small length scales starting from a few millimeters where surface tension forces are dominant compared with other internal viscous effects, resulting in bigger impact on the properties and shape of the flow. This phenomena is observed in many naturally occurring systems such as the vaporization flow in the xylem tissues in plants and in the wetting of water drops on leaf surfaces, while in technological and industrial applications can be found in processes such as in inkjet printing, surface coating, microfluidic devices and 3D printing. The above have augmented vast research interest and various numerical models and methods have been proposed to describe and accurately predict the formation and evolution of flow of surface tension effects to flows profiles.

The modelling of surface tension dominated flows requires the accurate prediction of fluid-fluid and solid-fluid interactions. Multicomponent flows in multiphase are experience in many applications, with their mathematical continuum-scale modelling been difficult by the strong non-linearity of the Navier-Stokes describing the flow [147]. Take for instance, the interaction between the fluid-solid molecules present additional challenges due to the inability for the standard (no slip) boundary conditions in addition with the Navier-Stoke equation not able to describe the molecular interaction during the fluid dynamics. Therefore, in dealing with solid-fluid interaction the analysis need to be simplify, but should not lose the important physics in the original

problem. To do this, the Lagrangian particle based method, known as Smoothed Particle Hydrodynamics (SPH), will be one sure way of achieving this. This SPH method is represented by a set of particles (Lagrangian particle method), which keeps individual material properties and also behaves according to the governing conservation equations. SPH was first developed to simulate astrophysical problems and up to date it has been widely studied and extended to dynamic fluid flows. The meshless nature of this approach makes it ideal for solving flow phenomena with complex geometries and moving interfaces.

1.1 Aims and objective

Our present study aim to refine the potential for modelling fluid droplet with the well known Lagrangian based method of SPH to eliminate the difficulties posed by the traditional grid based lubrication approximation technique. Despite that this area of simulating contact angle in SPH is relatively new, there are quite a few approaches which has been adapted, but not generally accepted. The major objectives of our study are given below:

• An efficient SPH solver should be developed from scratch in C++ programming language.

• The solver needs to be validated and benchmarked against existing numerical and analytical results before implementation.

• The developed SPH solver need to handle the modelling of the surface tension for single-phase in two-dimensions.

• Include correct prediction of curvature computation for single-phase in two-dimensions.

• For the first time the use of disjoining pressure approach to control the contact angle in SPH.

• For the first time the use of Contact Line Force (CLF) approach for single-phase in SPH.

Chapter 2

Literature review

2.1 Surface Tension Literature review

The surface tension phenomenon was introduced in the beginning of the 19th century by Pierre Simon de Laplace and Thomas Young. Fluid molecules inside the fluid are in absolute equilibrium due to the attractive forces among the all neighbor molecules. The attractive forces are not balanced for the molecules at the surface of the fluid due to absence of the same type molecules at the other side of the interface and this produces surface tension (see Figure 2.1). This is the main factor that liquids tune their shapes in order to minimize surface area, for instance, water forms spherical droplet in the absence of external forces (see Figure 2.2).

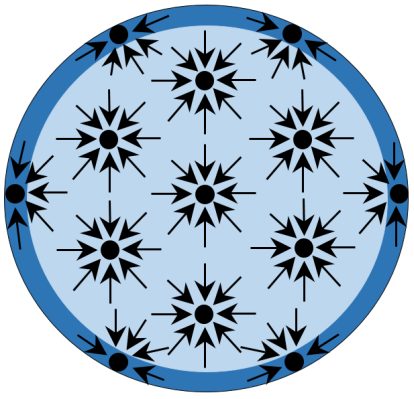


FIGURE 2.1: Assymetric surface tension effect at the interface of a fluid droplet

FIGURE 2.2: Perfect spherical droplet [42].

The surface of liquid, where the surface tension phenomenon exists, is a physical interface or a thin boundary layer between liquid and gas (for example air) with thickness of, approximately a few molecular diameters [55]. Surface tension, σ, can be described as a force per unit length with SI unit of N/m. Consider a rectangular metal frame as shown in Figure 2.3. Movable part is free to move in either directions, as shown in Figure 2.3. The metal frame is dipped in water, which forms a water film over the frame. The water film drags the movable bar, trying to decrease the surface area, because of the surface tension. The surface tension force acting on the movable

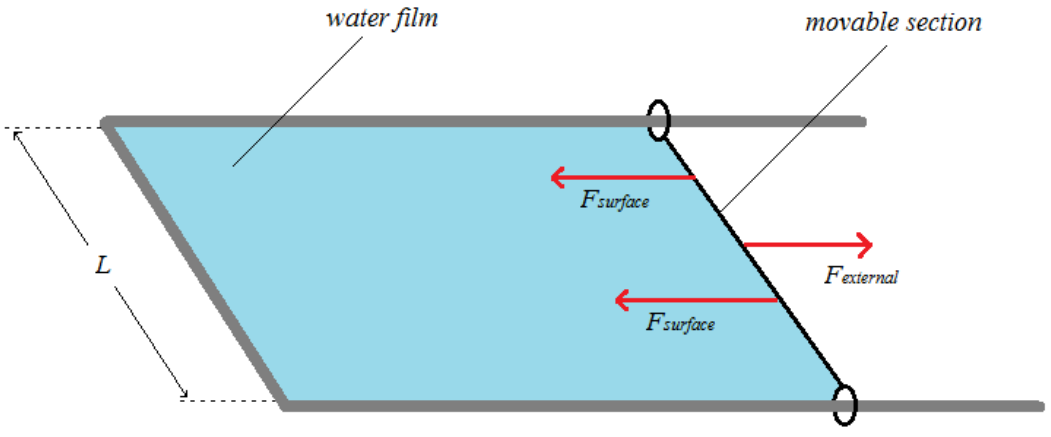


FIGURE 2.3: Demonstration of the surface tension force with a rectangular metal frame

bar is written as:

$$F = 2\sigma L \tag{2.1}$$

where the factor 2 explains the existence of two surfaces as shown in Figure 2.4

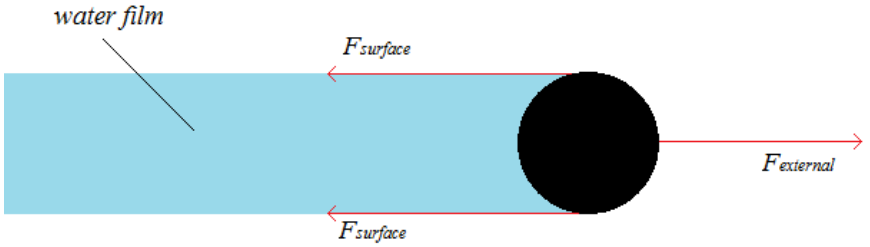


FIGURE 2.4: Water film with two surfaces.

More complex water/soap films can be achieved using different complex frames such as a hanging chain, a cubic shape and a spiral as shown in Figure 2.5. Phenomena that helps insects to walk on water is also surface tension force as illustrated in Figure 2.6a. These insects are walking on the surface of the water without getting wet, because their weight is balanced with the surface tension force directed upward as shown in Figure 2.6b.

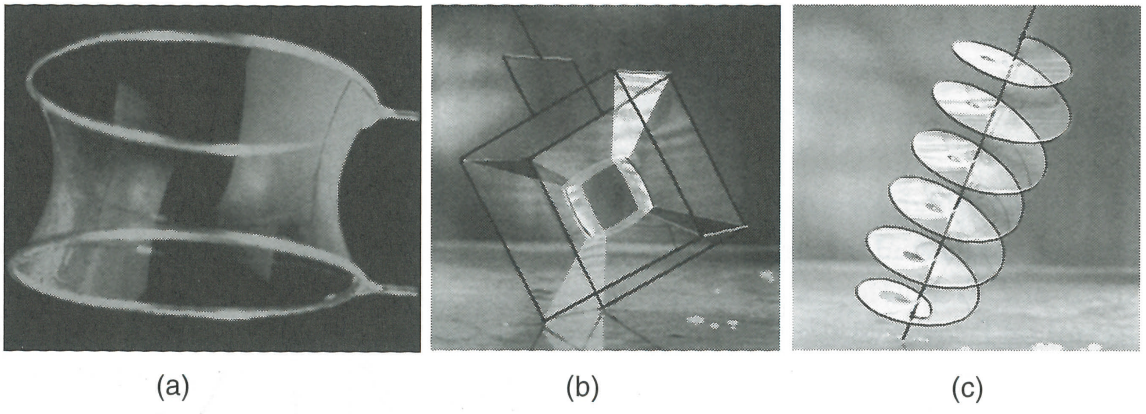


FIGURE 2.5: Soap films: a) hanging chain; b) cubic shape; c) spiral [42]

Logically, following the idea of surface tension, σ, as a shrinking interface, it must create a difference in pressure, $\triangle p$, between the liquid droplet and the surrounding environment. Figure 2.7 shows the force due to pressure difference, $\pi r^2 \triangle p$, and the force due to surface tension, $2\pi r\sigma$, along a circumference of circle of a spherical droplet.

The equilibrium condition becomes:

$$2\pi r\sigma = \pi r^2 \triangle p \tag{2.2}$$

or

$$\triangle p = \frac{2\sigma}{r} \tag{2.3}$$

This equation is a particular case of the widely known equation of Young and Laplace

$$\triangle p = \sigma \left(\frac{1}{r_1} + \frac{1}{r_2} \right) \tag{2.4}$$

where r_1 and r_2 are the principle radii of curvature of the surface.

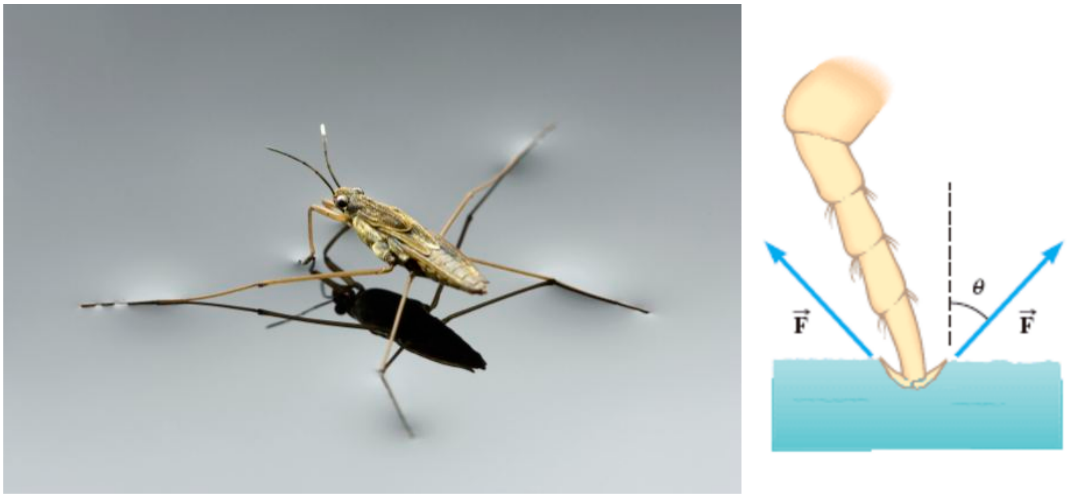


FIGURE 2.6: a) The surface tension force acting on the one foot of an insect. b) A water strider walking on the surface of the water without sinking with the help of the surface tension force directed upward.

In another word, the above equation means that the smaller the drop, accordingly, the higher internal pressure. This can be confirmed with two bubbles. By joining two different sized bubbles as shown in Figure 2.8, we can see that the smaller bubble becomes empty by discharging itself into the bigger bubble. This phenomenon can be seen in an emulsion of oil in water where smaller drops vanishes in bigger drops due to this higher pressure becoming thermodynamically unbalanced.

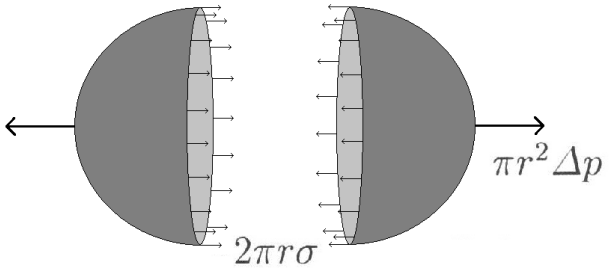


FIGURE 2.7: The surface tension force along a circumference of circle of a spherical droplet is $2\pi r\sigma$. This force is balanced with the force resulting due to the pressure difference $\pi r^2 \triangle p$, since the droplet at equilibrium state.

The surface tension of liquids depends on temperature of liquid since the surface tension arises from intermolecular attractive forces which in its case depends on the temperature. Figure 2.9 shows that the surface tension of water against air decreases with increasing temperature [153]. Details and relations for the surface tension, σ, dependence of temperature can be found in the literatures [3, 26].

The surface tension of medical liquids is essential for medical applications such as the surface tension of human blood (between $55.5 \cdot 10^{-3}$ N/m and $61.2 \cdot 10^{-3}$ N/m) and saliva ($53 \cdot 10^{-3}$ N/m) [5, 63]. Also it is an important property in refining and melting in liquid metal processing procedures such as sintering, casting and brazing because these processes are directly related with mechanical and thermodynamic properties of liquid metals [3, 26, 153]. Table 2.1 shows the surface tension values for some molten metals at their melting temperature [76].

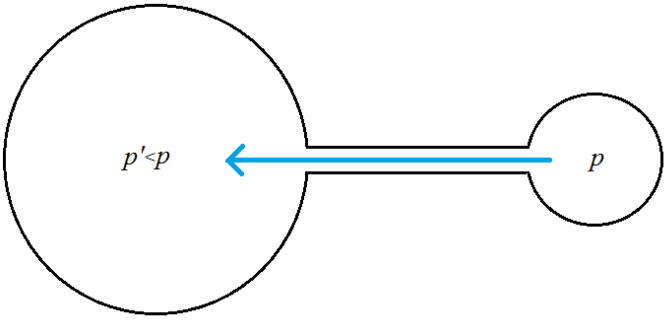


FIGURE 2.8: Large bubble absorbs small bubble because of the low pressure compared to the pressure of the small bubble.

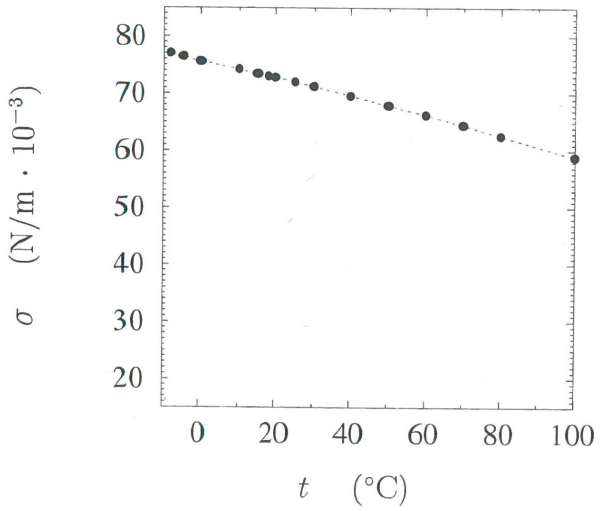


FIGURE 2.9: The temperature dependence of the surface tension of water against air [55].

Metal	σ (N/m)
Caesium	$70 \cdot 10^{-3}$
Platinum	$195 \cdot 10^{-3}$
Gold	$197 \cdot 10^{-3}$
Mercury	$498 \cdot 10^{-3}$
Aluminium	$914 \cdot 10^{-3}$
Silver	$966 \cdot 10^{-3}$
Titanium	$1650 \cdot 10^{-3}$
Iron	$1872 \cdot 10^{-3}$
Tunsten	$2500 \cdot 10^{-3}$

TABLE 2.1: Surface tension values for some molten metals at their melting temperature

2.1.1 Liquid-Liquid and Liquid-Solid Interfaces. Wetting

In previous chapter, surfaces between liquid and surrounding gas have been explained. Interfaces between liquid and liquid or between a solid and a liquid are also very

essential phenomenon. For example, lets consider an oil lens on water as shown schematically in Figure 2.10

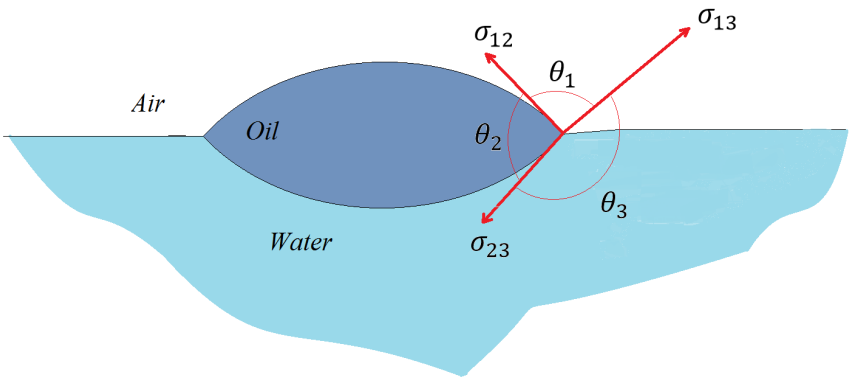


FIGURE 2.10: Oil lens on water surface

The surface tensions per unit length along the contact line of the three-phase are denoted as σ_{12}, σ_{13} and σ_{23} with angles θ_1, θ_2 and θ_3 between them. Equilibrium among these surface tension forces can be achieved if only the following condition is true

$$\frac{\sin\theta_1}{\sigma_{12}} = \frac{\sin\theta_2}{\sigma_{23}} = \frac{\sin\theta_3}{\sigma_{13}} \tag{2.5}$$

This equation is coming from the application of the law of sines to the triangle created by the surface tension forces at the contact line.

To consider the interaction between a liquid and a solid, lets assume that there is a liquid drop placed on a smooth solid substrate with a contact angle θ between the liquid and the smooth solid substrate as shown in Figure 2.11. In general, it is believed, that Young's relation applies which is written in the form

$$\sigma_{13} = \sigma_{23} + \sigma_{12}\cos\theta \tag{2.6}$$

Usually Equation (2.6) is obtained from the equilibrium condition between the horizontal components of the forces per unit length along the contact line of the three-phase. Objections against the Young's equation can be found in Bikerman [26] with

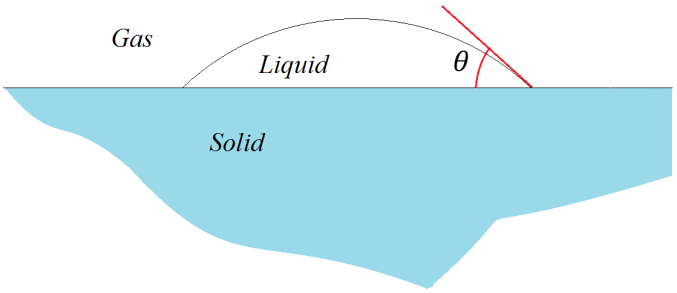


FIGURE 2.11: Liquid drop on smooth solid substrate

a serious discussions. Contact angles between the three-phases liquid, gas and solid depend on the chemical and physical characteristics of each phase.

Depending on the contact angles the expressions wetting (total wetting) and non-wetting (partial wetting) are used in different applications [25, 106]. In general, a spreading (contact angle $\theta \approx 0$) phenomenon of a liquid on a solid or liquid substrate is known as a wetting, total wetting or full wetting. Non-wetting occurs if the contact angle is greater than 90 degree. The wetting is of enormous significance in numerous industrial applications and in daily life, some of them are listed below:

- construction (protecting metals and concretes from water)
- glass (anti-frosting)
- automobile industry (processing tyres against slippery roads, processing glasses against dewetting of water)
- soil science (absorption of liquids by porous media)
- chemical industry (coating, inks, paints)
- insects walking on water surfaces
- tears in eyes
- climbing liquid in plants

Studying the wetting phenomenon in details lets us to describe why water drop spreads easily on glass but not on a metal. Better understanding of wetting phenomenon

enables us to adjust a surface to convert a wettable substrate into a non-wettable or in the other way around. For example, by depositing a thin fluorinated layer on glass, you can turn wettable glass into non-wettable glass like Teflon.

2.1.2 Surface Tension in Experiments and Numerical Simulations

Drop impact onto liquid and solid surfaces is ubiquitous in the fields like meteorology, spray cooling, quenching and painting, combustion engines, ink-jet printing, and annealing, etc. From a scientific point of view, it is known that the drop impact was first investigated by Worthington [157–159]. Since then different impact conditions such as shallow or deep liquids, low or high speed and cold or hot surfaces have been studied and with various articles published. It is known that depending on these conditions, various phenomena can arise: droplet bouncing can be observed [53, 70, 100, 130], a so-called Worthington jet may appear [44, 138], the drop may splash with a crown [83, 162], the drop spreading may occur after impact onto the solid substrate [31, 56, 57, 59, 71], and etc.

Furthermore, depending on the flow rate, different flow regimes can be formed, for example, Figure 2.12 shows experimental photos done by Schmuki and Laso [134] for four different flow regimes. In their work, different flow regimes using narrow tube to bring a liquid at a constant flow rate from the top of the inclined surface. "Sliding drops" with equal distance from each other was observed at the lowest flow rate as shown in Figure 2.12a. The sliding drops start to join and formed a "straight rivulets" when the flow rate was increased as shown in Figure 2.12b. The straight rivulets lose its form and become windy rivulets when you keep increasing the flow rate as shown in Figure 2.12c which finally formed a "film" with waved surface at the highest flow rate as shown in Figure 2.12d.

The physical governing equations of the fluid dynamics can be described by the Lagrangian and the Eulerian descriptions. Depending on types of problem, both of

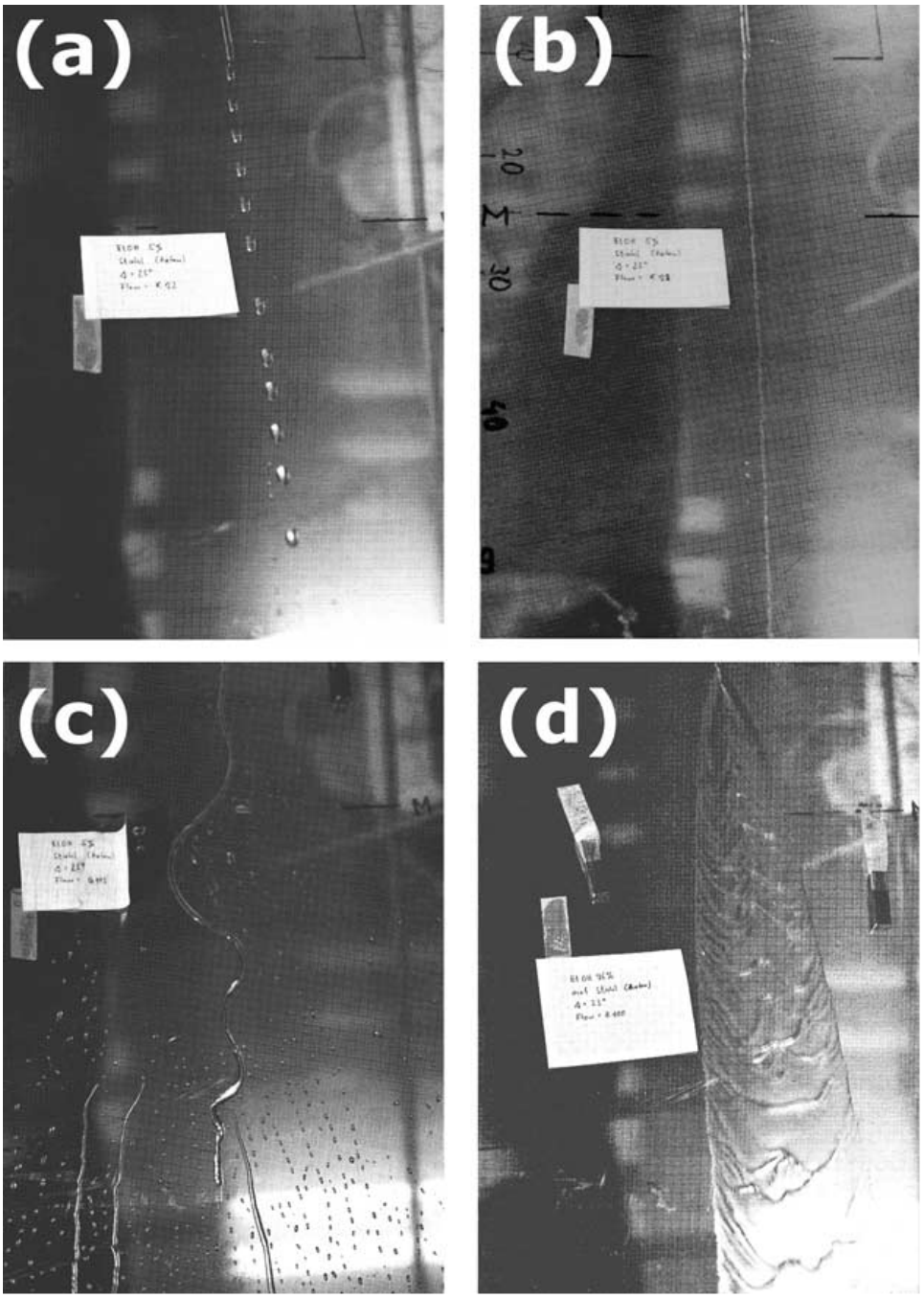


FIGURE 2.12: Experimental photos observed by Schmuki and Laso [134] to study different flow regimes: a) sliding drops, b) straight rivulets, c) windy rivulets, and d) film

these descriptions can be implemented in the Computational Fluid Dynamics (CFD). Some brief explanation about these two descriptions will be explained and compared in the next section. In this section, implementation methods of the surface tension phenomenon in the numerical simulation will be discussed in details.

Modelling surface tension dominated flows involve dynamic tracking of the interface caused by the asymmetric cohesion forces of fluid on the free-surface (see Figure 2.1). The accurate prediction of surface tension dominated flows with free-surfaces, such as those observed in droplets, has been a bane in traditional CFD methods. Mainly, numerical simulations can be efficient and successful in capturing the major results of fluid-solid interactions, while experimental investigation needs more supplies and time. Widely used and accepted numerical simulation approaches for simulating fluid-solid interactions are Eulerian based. For example, the impact of single drop onto liquid films taking in account the surface tension was numerically simulated with boundary element method [40, 154], finite element method [30], level set method [163], Lattice Boltzmann method [124], and volume of fluid (VOF) [35, 149]. Fawehinmi *et al.* [51] used commercial CFD packages, such as FLOW-3D and CFX which are based on the VOF method to investigate the effect of viscosity on the dynamics of drop formation in the dripping mode. Gaskell [60] simulated a range of droplet spreading flows on different topographic substrates with various wetting conditions using a full approximation storage (FAS) Multigrid method.

The present work explores the use of fully Lagrangian methods, such as Smoothed Particle Hydrodynamics (SPH), to reliably capture and model the complex dynamic behaviour of surface tension dominated free-surface flows. The SPH methodology was originally developed independently by Gingold, Monahgan [62] and Lucy [97] for astrophysical phenomena at the hydrodynamic scale. More recently, SPH has found numerous applications in modelling fluid flow problems including those with moving interfaces [142]. The major advantage of SPH over the Eulerian approach is that the fluid, represented by particles, is naturally suitable for complex geometries and problems arising from free-surface flows.

Several approaches to accurately model surface tension has been reported (Brackbill *et al.* [29], Nugent, Posch [128] and Tartakovsky, Meakin [145]) and they can be broken down to two major methodologies. The Continuous Surface Force (CSF) considers the curvature of the surface of the fluid (Brackbill *et al.* [29]) and was popularised by Muller *et al.* [125], who proposed a single-phase method to track and model surface

tension on droplets in SPH with the aid of different kernels. While the approach is fast, with applications in computer graphics, it suffers from accuracy that is necessary to solve practical and realistic physical problems. Das and Das [39] studied multiphase flow of drops moving down an inclined plane. They employed the CSF method for their surface tension model while proposing a diffused interface model for tracking the motion of the contact line. In order to resolve clumping and particle penetration issues, they employed an inter-particle Leonard-Jones force [93]. The second approach focuses on an Inter-particle Interaction Force (IIF) which was employed by Meakin and Tartakovsky [101] in a porous media problem, that involves the combination of repulsive and attractive forces between the fluid molecules, thereby causing surface tension forces to exist even within the fluid. These unbalanced forces within the bulk of the fluid make the droplet very sensitive, especially at the contact line with small contact angles. While the surface tension model uses IIF, the particle pressure is calculated using the Van der Walls equation of state where it requires the use of three control variables [68, 128]. Akinci *et al.* [4] investigated using a multi-phase approach where they proposed a combine surface tension model comprising of the cohesion and surface area minimization terms to model the fluid and air particles while an adhesion model is introduced for fluid and solid particles. This resulted in a method that is computational expensive in terms of computational cost and time.

In the present work, a single-phase modified CSF approach is developed to accurately model surface tension dominated flow of droplets. The proposed methodology introduces the use of a modified curvature model, where the formulation will be verified against known results numerically and those in literature. Both CSF and IIF approaches are employed to investigate the case of an oscillating droplet and droplet contact angle with real fluid properties; the former aims simplify and eliminate the use of tuning parameters that may result in unphysical and unrealistic interactions. Finally, the IIF approach was chosen to simulate and study the droplet contact angle hysteresis.

2.2 Numerical Simulation

In modern engineering and science, numerical simulation has taken significant role in solving complicated problems by the help of the powerful computers. Numerical simulation helps to convert the main features of a problems in physics into a discrete form of mathematics and then rebuilds and solves the task using computer, and the requirements of the experts are mostly satisfied with the solution. One of the advantages of this numerical method is that it solves the problems with less hypothesis in all its details using modern computers rather than the traditional methods with more approximations and assumptions.

In modern scientific research the numerical modelling has become an alternative method, instead of doing dangerous, time-wasting and expensive experiments for people [93]. In terms of providing complete and clear data, the numerical modelling methods are usually more convenient than the traditional methods like observing and measuring directly or with other difficulties.

Numerical modelling with powerful computers is a very important in delivering a verification assumptions, presents understandings to the physical investigations and also helps to explain or even to discover new ideas. As was mentioned in Liu's book [93], numerical simulation plays important role as a link between experiment and theory as shown in Figure 2.13.

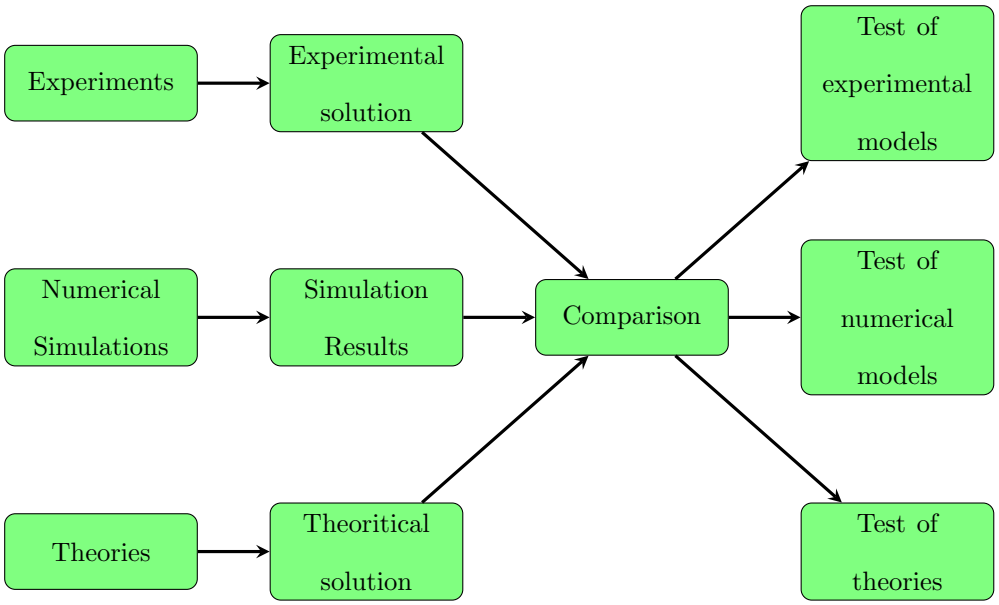


FIGURE 2.13: The numerical simulation plays role of a connection between theories and experiments and helps to understand and/or to explain the physical investigations.

2.2.1 Mechanism of solving numerical simulations

To overcome with a practical problems, numerical simulations pay attention to some processes. Basically, according to Liu [93], there are some main and required steps in the process, as illustrated in Figure 2.14. Mathematical models are formed with reasonable suppositions and simplifications from the discovered physical experience. Usually, mathematical models are stated as governing equations with some appropriate conditions such as initial conditions (IC) or/and boundary conditions (BC). The governing equations can be in the form of partial differential equations (PDE), ordinary differential equations (ODE) and also can be in the form of different physics law equations. To establish the field functions (e.g. mass, velocity, pressure) in time or space, initial or/and boundary conditions are required.

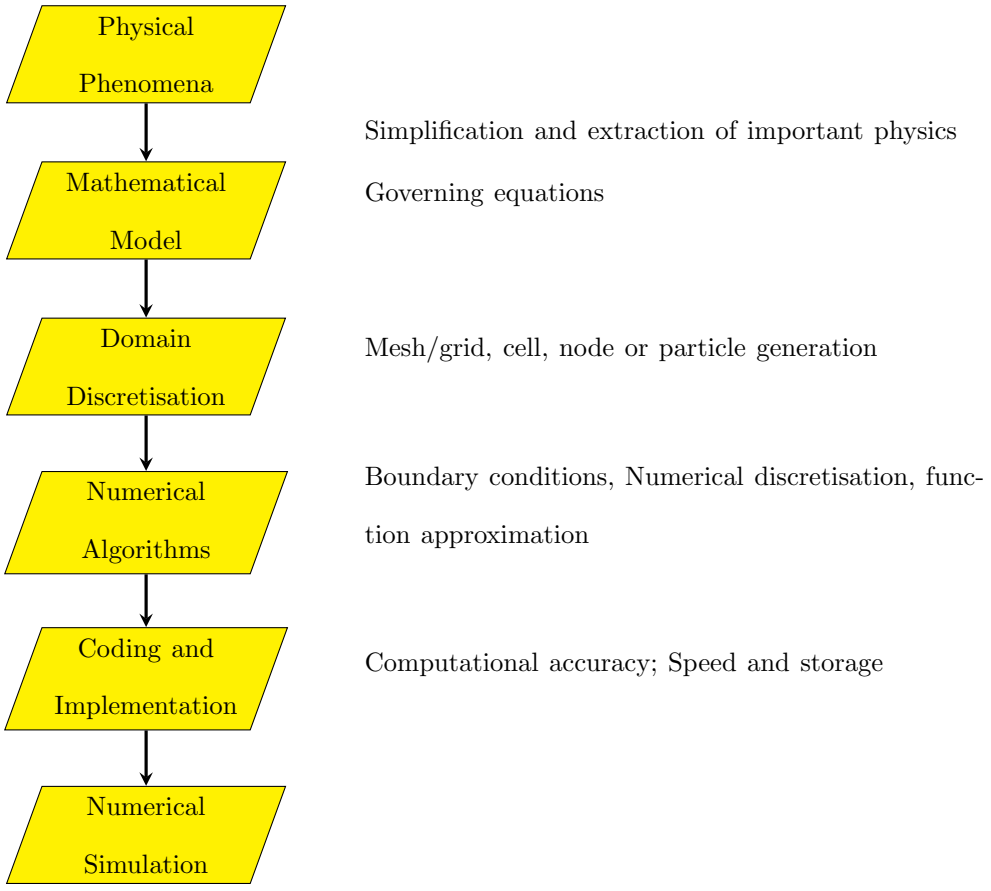


FIGURE 2.14: Steps in the way of getting numerical simulation

Dividing the required geometry of the problem domain into discrete components is necessary for solving the governing equations numerically. The domain discretization methods can be various depending on numerical methods. The computational frame to the numerical approximation is formed by the representation of a required domain with a limited number of elements which is known as domain discretization. Traditionally, the computational frame is in the form of the grid or mesh, which is in the form of grid nodes or grid points for approximation of the domain. The grid nodes are connected by some kind of nodal connectivity and the positions where the physical field functions are measured. Following the connectivity, a mesh is formed by the connection of the grid nodes. The mesh patterns and the mesh cell size are deeply affected to the accuracy of the numerical approximation.

The tool for changing the continuous forms of the governing equations to discrete representations is known as numerical discretization. The numerical discretization

technique and the domain discretization technique are both closely related to each other. As was described in Liu [88], on a base of function approximation the numerical discretization is developed. The equations in physics can be rewritten in the form of algebraic equations or ODE, that can be established by the already known numerical methods after numerical and domain discretization. The weak or strong form formulations may be used in the way of defining the algebraic equations or ODE [88]. The strong and weak form formulations can be combined together to get the best advantages of these forms of formulation as it has been described by Liu and Gu [92]. The improvement of the meshfree weak-strong technique developed from the combination of the both formulations.

Converting the numerical algorithms and the domain decomposition into a computer code in certain programming languages is the way of working of a numerical simulation. The efficiency (storage and speed) and accuracy are two main considerations in coding a computer program. Other considerations include usage (using, modifying and reading), validity of the code etc. The programing code must be confirmed to theoretical solutions, or the known results from existing verification tests before performing a practical numerical simulation.

The governing equations are obtained from the physical conservation laws, such as the momentum, energy or mass and also these variables should be conserved. The behavior of the fluid system is determined the above mentioned principles, with other details of the medium and conditions at the boundary. The governing equation is the fundamental physical principles which are conveyed in the form of some key algebraic equations. In various cases the governing equations occur as the PDE.

It is very hard to get analytical solution of partial differential or integral equations, except for a few cases. Computational fluid dynamics (CFD) considers methods of approximating the integrals which are generally in the form of mathematical summation. The above mentioned approximation in the form of mathematical equations which could be evaluated to get numerical value for field variables (pressure, velocity, mass) at discretized points (Figure 2.15). The aspects listed below are the parts of a

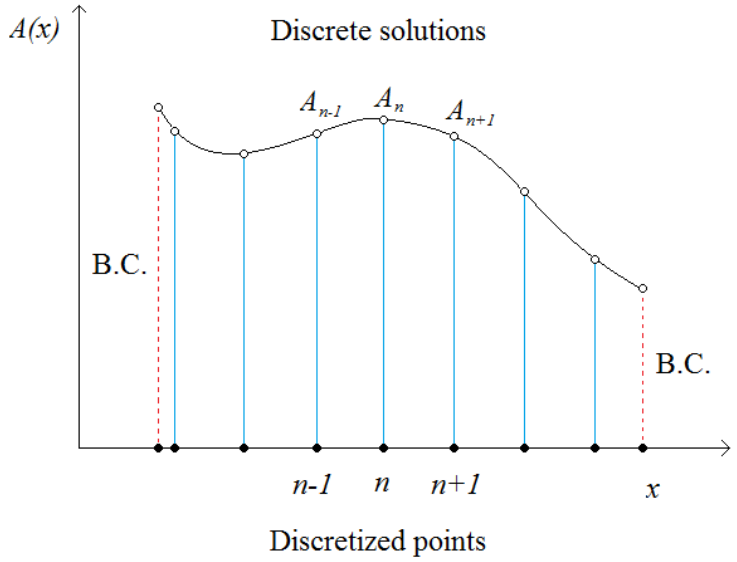


FIGURE 2.15: Domain and numerical discretization of a $f(x)$ function in 1D.

common numerical simulation for the CFD issue.

· Boundary Condition

· Governing Equation

· Domain and Numerical Discretization Methods

· Numerical method for solving the ODE or mathematical equations.

The physical governing equations are described by the Lagrangian (material description) and the Eulerian (spatial description) descriptions. Depending on types of problem, both of these descriptions are used in numerical methods. Some brief explanation about these two descriptions are given in the next section.

2.3 Grid-based Methods

The Lagrangian and the Eulerian descriptions can be used to explain the physical governing equations. The Lagrangian description is represented by the finite element method (FEM) and is a material description [94, 166]. The Eulerian description

is represented by the finite difference method (FDM) and is a spatial description [6, 73, 156].

The Lagrangian grid and the Eulerian grid are widely used as the standard technique in numerical methods depending on types of problems, and shortly described in the following.

Lagrangian Grid

In the Lagrangian grid, the grid is attached (fixed) on the material, hence it moves with the material (see Figure 2.16a) [94]. By the movement of the mesh cells also mass, momentum and energy are transported with them.

Here are the some advantages.

1. The code is simple, it should be faster because no convective term exists in the PDE.
2. The time history the variables can be tracked because the grid is attached on the moving material.
3. By placing grid nodes along boundaries, the boundary conditions at moving boundaries and free surfaces are determined just by the movement of grid nodes.
4. Complex geometries can be easily treated by irregular mesh.

These advantages of the Lagrangian methods makes it choice of many researchers for their works.

Eulerian Grid

In contrast to the Lagrangian grid, the Eulerian grid is attached to space. The considered object passes the fixed mesh cell in the grid (Figure 2.16b). All grid nodes

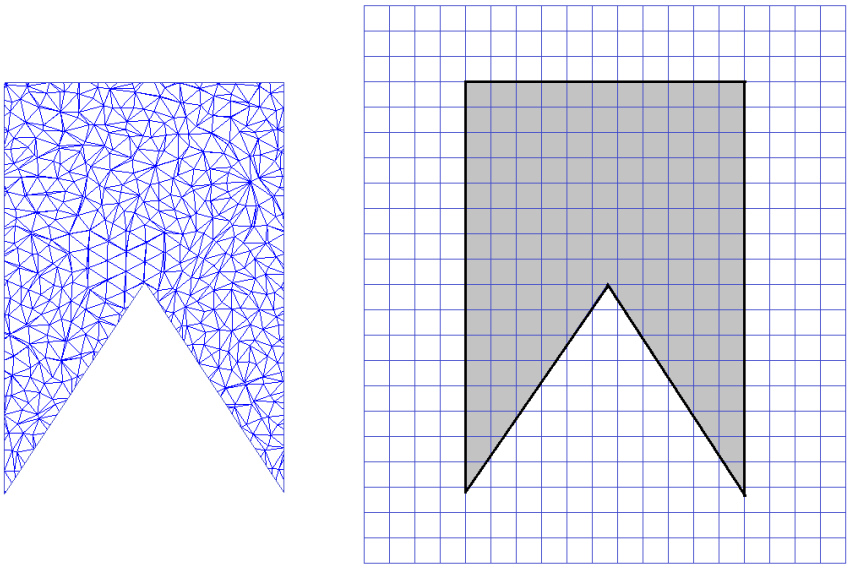


FIGURE 2.16: a)(left) Lagrangian grids, the cells move with the material. b) (right) Eulerian grids, the grid is fixed in space, so it doesn't deform and move with time.

stay fixed in space not depending on time while the materials are moving across the mesh. The volume and shape of the mesh cell are not changing during the process of the computation.

The large deformations in materials are not affecting to the mesh's shape or volume (no deformation).

Liu [93] compared some advantages and disadvantages of the Lagrangian and Eulerian methods in his book as shown in Table 2.2.

2.4 Meshfree Method

The main aim of the meshfree methods is to insure stable and accurate numerical solutions for integral equations with all kinds of boundary conditions with a set of distributed particles (or nodes) without using mesh that allows the connectivity of these particles. The most meshfree methods have some common features, but are different in the terms of the implementation process and function approximation.

	Lagrangian methods	**Eulerian methods**
Grid	Attached on the moving material	Fixed in the space
Track	Movement of any point on material	Mass, momentum, and energy flux across grid nodes and mesh cell boundary
Time history	Easy to obtain time-history data at a point attached on materials	Difficult to obtain time-history data at a point attached on material
Moving boundary and interface	Easy to track	Difficult to track
Irregular geometry	Easy to model	Difficult to model with good accuracy
Large deformation	Easy to handle	Difficult to handle

TABLE 2.2: Comparison of the Lagrangian and Eulerian methods

Smoothed Particle Hydrodynamics (SPH) [62, 97], as a particle and meshfree method, was originally used for modelling astrophysical problems, and later widely applied for applications to problems in fluid dynamics. The SPH method is the main type of particle methods, and have been applied to many commercial codes.

In Table 2.3 given some meshfree methods and methods of approximation with references in chronological order.

2.5 The SPH method

The Smoothed Particle Hydrodynamics (SPH) method is represented by a set of particles (Lagrangian particle method), which keeps individual material properties and also behaves according to the governing conservation equations. SPH was first developed to simulate astrophysical problems and up to date it has been widely studied and extended to dynamic fluid flows. The SPH was extended for dynamic fluid flows which are generally in partial differential equations form of field functions. Particles movement depend on physical quantities which are obtained from the nearest neighbour particles in terms of the weighting function which is solved from its compact

Methods	**Methods of approximation**	**References**
Smoothed Particle Hydrodynamics (SPH)	Integral representation	Lucy [97]; Gingold and Monaghan [62], etc.
Finite point method	Finite difference representation	Liszka and Orkisz [87]; Onate *et al.* [129], etc.
Diffuse element method (DEM)	Moving least square (MLS) approximation Galerkin method	Nayroles *et al.* [127]
Element free Galerkin (EFG) method	MLS approximation Galerkin method	Belytschko *et al.* [12–14] etc.
Reproduced kernel particle method (RKPM)	Integral representation Galerkin method	Liu *et al.* [95, 96] etc.
HP-cloud method	MLS approximation, Partition of unity	Duarte and Oden [45] etc.
Free mesh method	Galerkin method	Yagawa and Yamada [160, 161] etc.
Meshless local Petrov-Galerkin (MLPG) method	MLS approximation Petrov-Galerkin method	Atluri and Zhu [9]; Atluri and Shen [8]; etc.
Point interpolation method (PIM)	Point interpolation, Galerkin method, Petrov-Galerkin method	Liu and Gu [89–91]; Gu and Liu [66, 67]; Liu [88]; Wang and Liu [152].
Meshfree weak-strong form (MWS)	MLS, PIM, radial PIM, Collocation plus Petrov-Galerkin	Liu and Gu [92]; etc.

TABLE 2.3: Meshfree method in chronological order

support value.

2.5.1 Advantages and limitations of SPH

Here, some advantages of the SPH method over the Eulerian methods are given as follows:

• Complex geometries can be easily programmed without need of any complicated algorithms as it can be seen in FEM

• Empty regions of the problem domain are not considered at all, the computational effort is concentrated only in regions where the material is located by saving the time needed for computation

• The code algorithms can be easily modified depending on the new physical processes

• With SPH method, free surface problems such as droplet or breaking waves provides better results and easy to apply

Some limitations of the SPH method are given as follows:

• Boundary conditions should be applied very carefully and penetration of the fluid particles into the wall particles should be avoided

• The computational cost might be very expensive for the problems with high resolution due to the number of the particles and also the time integration scheme

• Sometimes, the distribution of the particles may not be spaced homogenously and special techniques need to be involved

2.5.2 Applications of SPH

The SPH method is represented by a set of particles (Lagrangian particle method), which keeps individual material properties and also behaves according to the governing conservation equations. SPH was first developed to simulate astrophysical problems and up to date it has been widely studied and extended to dynamic fluid flows [62, 97]. One of the SPH methods advantages is the adaptive nature. In addition to the adaptive nature, other attractive feature of the SPH method is the harmonic combination of the particle approximation with Lagrangian formulation. The other reason of the SPH has been applied successfully to a broad range of problems is that it is relatively easy to consider complex physical effects into the SPH formalism. Examples of SPH application are found in a wide variety of fields:

ASTROPHYSICS	Stellar collisions [16, 18, 49, 50, 54, 116]
	Moon formation and impacts problems [17]
	Cloud fragmentations and collisions [47]
	Supernovas explosion [72, 126]
	Cosmology [48, 137]
	Collapse and formation of galaxies [22–24, 114]
	Evolution of the universe [108]
MAGNETO - HYDRODYNAMICS	Magnetic collapse of gas clouds [69]
	Alfvenic waves propagation [131]
	Development of expansive wave in a magnetic cloud [139]
SOLID MECHANICS	Impact problems [77, 84, 85]
	Fractures simulation [19]
	Impacts of solids simulation [20]
	Study of brittle solids [21]
	Metal forming [27]
FLUID DYNAMICS	Multi-phase flows [111]
	Heat conduction [33]
	Underwater explosions [141]
	Gravity currents [109, 110]
	Free-surface flows [112, 113, 117]

TABLE 2.4: Application of the SPH method in a different fields

Chapter 3

SPH Methodology

3.1 Basic SPH formulation

In general, the SPH methodology is described by two main parts. The first is the kernel approximation (integral representation) of field functions. The other is the particle approximation.

3.1.1 Kernel Approximation

In the kernel approximation step, the integral representation of the function is obtained by the integration of an arbitrary and a smoothing kernel function. This is approximated by summing up the values of neighbouring particles within the support domain.

The SPH kernel approximation method begins from the following integral equation of any quantity function:

$$A(\mathbf{r}) = \int_{\Omega} A(\mathbf{r}')\delta(\mathbf{r} - \mathbf{r}')d\mathbf{r}', \tag{3.1}$$

where A is a continuous function of the three-dimensional position vector $\mathbf{r}$, Ω is the volume of the integral that contains $\mathbf{r}$, $\mathbf{r}'$ is position vector of the neighbour point, $d\mathbf{r}'$

is an elementary volume and $\delta(\mathbf{r}-\mathbf{r}')$ is the Dirac delta function given by

$$\delta(\mathbf{r}-\mathbf{r}') = \begin{cases} 1 & \mathbf{r}=\mathbf{r}' \\ 0 & \mathbf{r}\neq\mathbf{r}'. \end{cases} \tag{3.2}$$

An integral representation of function $A(\mathbf{r})$ is obtained by replacing the Delta function $\delta(\mathbf{r}-\mathbf{r}')$ with a kernel smoothing function $W(\mathbf{r}-\mathbf{r}', h)$:

$$A(\mathbf{r}) \doteq \int A(\mathbf{r}')W(\mathbf{r}-\mathbf{r}', h)d\mathbf{r}' \tag{3.3}$$

where h is a smoothing length which specifies the size of the supporting domain of the kernel, shown in Figure 3.1.

As long as W is not the Dirac delta function, the integral representation can be an approximation via Equation (3.3). The kernel approximation operator is marked by the angle bracket <>, and Equation (3.3) is rewritten as:

$$A(\mathbf{r}) \cong <A(\mathbf{r})> = \int_\Omega A(\mathbf{r}')W(\mathbf{r}-\mathbf{r}', h)d\mathbf{r}' \tag{3.4}$$

The kernel should satisfy the following conditions:

$$\int W(\mathbf{r}-\mathbf{r}', h)d\mathbf{r}' = 1. \tag{3.5}$$

This is the normalization condition, also termed as unity condition because the integration of the smoothing function produces unity. The other condition is the kernel tends to a Delta function as $h \to 0$ in order to make sure that the approximation value tends the function value, i.e. $<A(\mathbf{r})> = A(\mathbf{r})$:

$$\lim_{h\to 0} W(\mathbf{r}-\mathbf{r}', h) = \delta(\mathbf{r}-\mathbf{r}'). \tag{3.6}$$

And also the kernel function should be an even function which means that particles at the same separation but different locations should have same influence on an interested

particle.

$$\int_{\Omega} (\mathbf{r}' - \mathbf{r}) W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}' = 0. \tag{3.7}$$

The last condition is the compact support condition:

$$W(\mathbf{r} - \mathbf{r}', h) = 0 \qquad \text{when} \qquad |\mathbf{r} - \mathbf{r}'| > kh, \tag{3.8}$$

where k defines the effective area of the smoothing function.

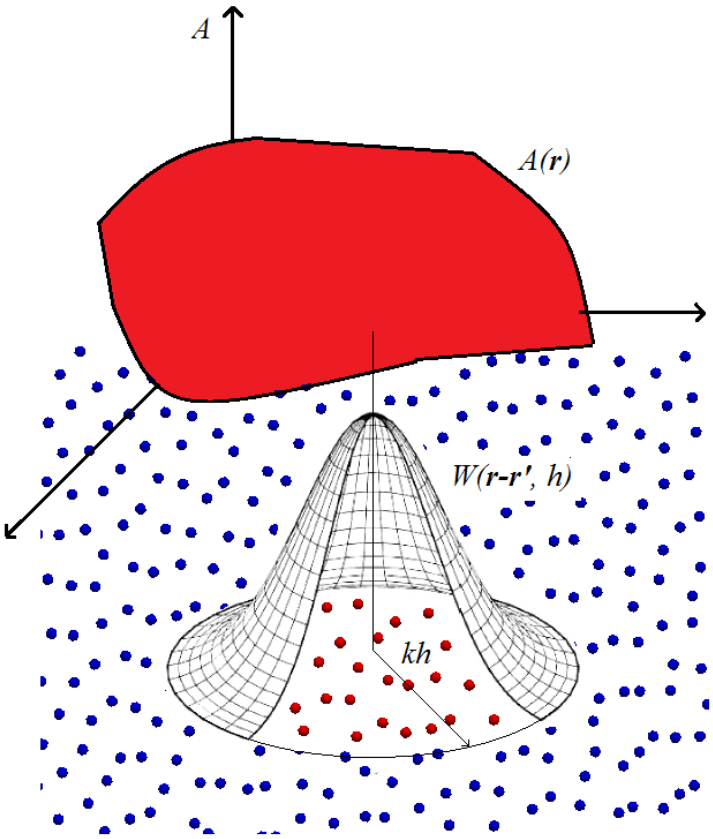


FIGURE 3.1: Kernel approximation of function A.

The error between Equation (3.1) and Equation (3.4) due to the approximation is second order accurate $O(h^2)$ [58, 116]. This error can be determined from the Taylor

expansion of $A(\mathbf{r}')$ about $\mathbf{r}$:

$$\begin{aligned} <A(\mathbf{r})> &= \int_\Omega [A(\mathbf{r}) + A'(\mathbf{r})(\mathbf{r}' - \mathbf{r}) + R((\mathbf{r}' - \mathbf{r})^2)]W(\mathbf{r} - \mathbf{r}', h)d\mathbf{r}' \\ &= A(\mathbf{r}) \int_\Omega W(\mathbf{r} - \mathbf{r}', h)d\mathbf{r}' \\ &+ A'(\mathbf{r}) \int_\Omega (\mathbf{r}' - \mathbf{r})W(\mathbf{r} - \mathbf{r}', h)d\mathbf{r}' + R(h^2) \end{aligned} \tag{3.9}$$

where R is remainder introduced in the approximation of A at point $\mathbf{r}$. Finally, including Equations (3.5) and (3.7) into Equation (3.9), yields:

$$<A(\mathbf{r})>= A(\mathbf{r}) + R(h^2) \tag{3.10}$$

So it can be seen from the above equation, the error in kernel approximation is second order accurate.

3.2 SPH derivative formulation

By substituting $A(\mathbf{r})$ with $\nabla \cdot A(\mathbf{r})$ in Equation (3.4), the approximation for the derivative $\nabla \cdot A(\mathbf{r})$ is obtained:

$$<\nabla \cdot A(\mathbf{r})>= \int_\Omega \nabla \cdot A(\mathbf{r}')W(\mathbf{r} - \mathbf{r}', h)d\mathbf{r}'. \tag{3.11}$$

In the right hand side of the above equation the divergence is taken with respect to the primed coordinate system:

$$\nabla \cdot A(\mathbf{r}')W(\mathbf{r} - \mathbf{r}', h) = \nabla \cdot [A(\mathbf{r}')W(\mathbf{r} - \mathbf{r}', h)] - A(\mathbf{r}') \cdot \nabla W(\mathbf{r} - \mathbf{r}', h). \tag{3.12}$$

Therefore, Equation (3.11) becomes:

$$<\nabla \cdot A(\mathbf{r})>= \int_\Omega \nabla \cdot [A(\mathbf{r}')W(\mathbf{r} - \mathbf{r}', h)]d\mathbf{r}' - \int_\Omega A(\mathbf{r}') \cdot \nabla W(\mathbf{r} - \mathbf{r}', h)d\mathbf{r}'. \tag{3.13}$$

The first term of the right hand side of the above equation can be rewritten as an integral over surface, S, of the domain of the integration, Ω:

$$<\nabla \cdot A(\mathbf{r})>= \int_S A(\mathbf{r}')W(\mathbf{r}-\mathbf{r}',h)\cdot \vec{\mathbf{n}}dS - \int_\Omega A(\mathbf{r}')\cdot \nabla W(\mathbf{r}-\mathbf{r}',h)d\mathbf{r}', \qquad (3.14)$$

where $\vec{\mathbf{n}}$ is the unit vector normal to the surface S.

For the case when the problem domain totally covers the support domain (see Figure 3.2a), the surface integral on the right hand side of Equation (3.14) vanishes, as long as the smoothing function W is having compact support. However, in the case when the problem domain does not totally cover the support domain (see Figure 3.2b), the surface integral is not zero anymore, because the smoothing function, W, is truncated by the problem domain. To avoid such truncation problems, corrections, such

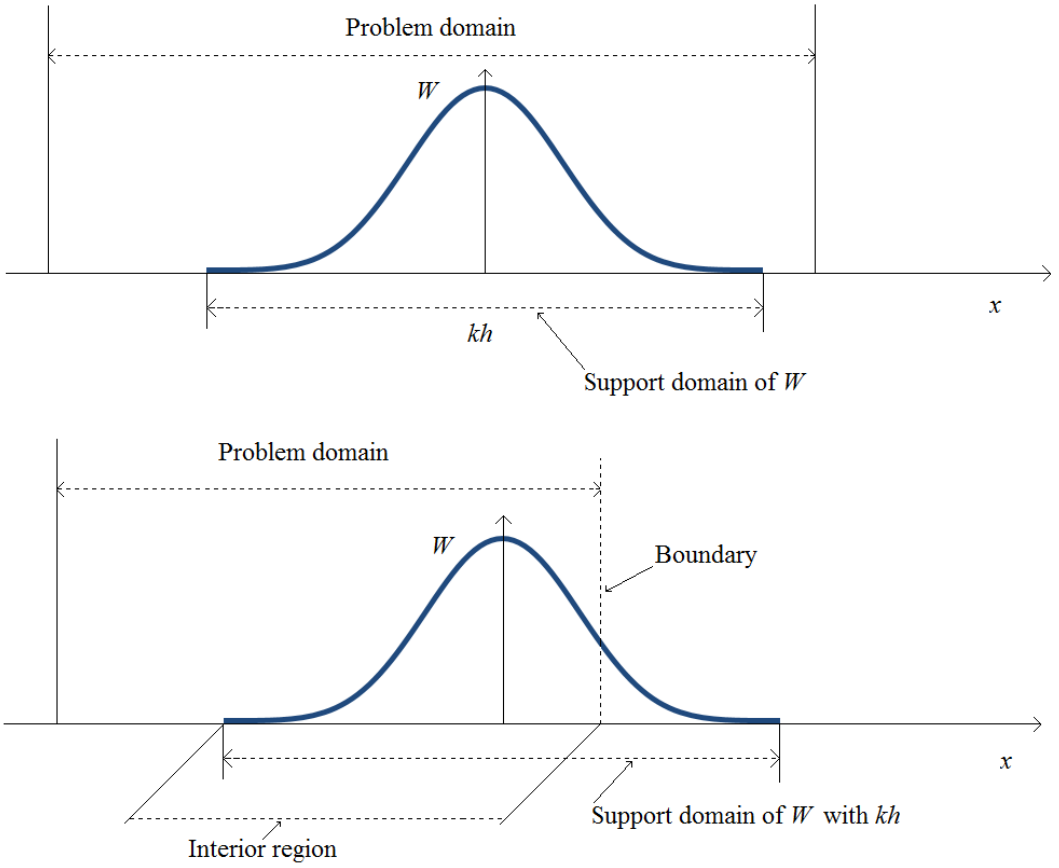


FIGURE 3.2: a)(Top) The surface integral on the right hand side of Equation (3.14) is zero b)(Bottom) The smoothing function W is truncated by the boundary, and the surface integral on the right hand side of Equation (3.14) is no longer zero.

as introducing the ghost particles, should be made to solve the boundary influences and the the surface integral can be treated as zero in Equation (3.14). Therefore, for those points whose support domain is inside the problem domain, Equation (3.14) is simplified as follows [93].

$$<\nabla \cdot A(\mathbf{r})>= -\int_{\Omega} A(\mathbf{r}') \cdot \nabla W(\mathbf{r}-\mathbf{r}', h) d\mathbf{r}'. \tag{3.15}$$

As it can be seen from above equation, the differential operation on a function is passed to a differential operation on the kernel function.

3.3 Particle Approximation Discretisation

The SPH method is represented by particles which carry individual mass. This can be solved using particle approximation discretisation. In the support domain, equation (3.3) can be converted to the discretized form by summing over all the particles as shown in Figure 3.3. Here, i is particle of interest and j's are it's neighbours including itself.

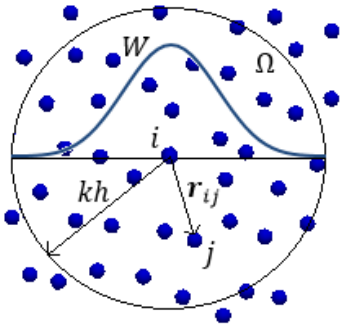


FIGURE 3.3: Particle approximations using particles within the support domain of the smoothing function W for particle i. The support domain is circular with a radius of kh

If the volume $d\mathbf{r}'$ at the location of particle j is replaced by the volume ΔV_j, then mass of the particle:

$$m_j = \Delta V_j \rho_j, \tag{3.16}$$

where ρ_j is the density of particle.

Equation (3.3) can be rewritten in the discretised particle approximation form:

$$\begin{aligned} A(\mathbf{r}) &= \int_\Omega A(\mathbf{r}')W(\mathbf{r}-\mathbf{r}',h)d\mathbf{r}' \\ &\cong \sum_{j=1}^{N} A(\mathbf{r}_j)W(\mathbf{r}-\mathbf{r}_j,h)\Delta V_j \\ &= \sum_{j=1}^{N} A(\mathbf{r}_j)W(\mathbf{r}-\mathbf{r}_j,h)\frac{1}{\rho_j}(\rho_j \Delta V_j) \\ &= \sum_{j=1}^{N} A(\mathbf{r}_j)W(\mathbf{r}-\mathbf{r}_j,h)\frac{1}{\rho_j}(m_j), \end{aligned} \tag{3.17}$$

or

$$A(\mathbf{r}) = \sum_{j=1}^{N} \frac{m_j}{\rho_j} A(\mathbf{r}_j)W(\mathbf{r}-\mathbf{r}_j,h). \tag{3.18}$$

The particle approximation for a function at particle i can be written as:

$$< A(\mathbf{r}_i) >= \sum_{j=1}^{N} \frac{m_j}{\rho_j} A(\mathbf{r}_j)\cdot W_{ij}, \tag{3.19}$$

where

$$W_{ij} = W(\mathbf{r}_i-\mathbf{r}_j,h). \tag{3.20}$$

Following the same way, the particle approximation for the spatial derivative of the function is:

$$< \nabla\cdot A_i(\mathbf{r}) >= -\sum_{j=1}^{N} \frac{m_j}{\rho_j} A(\mathbf{r}_j)\cdot \nabla W(\mathbf{r}_i-\mathbf{r}_j,h). \tag{3.21}$$

And, for a function at particle i:

$$< \nabla\cdot A_i(\mathbf{r}) >= \sum_{j=1}^{N} \frac{m_j}{\rho_j} A(\mathbf{r}_j)\cdot \nabla_i W_{ij}, \tag{3.22}$$

where

$$\nabla_i W_{ij} = \frac{\mathbf{r}_i - \mathbf{r}_j}{r_{ij}} \frac{\partial W_{ij}}{\partial r_{ij}}. \tag{3.23}$$

Taking in account that $\nabla_i W_{ij}$ is taken with respect to particle i, so the negative sign in Equation (3.21) is removed in Equation (3.22).

The summation in Equation (3.22) is executed over all neighbour particles in the supporting domain. It can be seen from Equations (3.19) and (3.21) that the particle approximation transforms the continuous integral representations of a function and also its derivatives to the discretised summations using an arbitrary set of particles. Actually, this approximation makes the SPH method simple.

3.4 Methods for the derivation of the SPH formulations

SPH formulation can be obtained for partial differential equations with the above mentioned technique of particle and kernel approximations. However, there are a few other ways to obtain SPH formulation exist to describe partial differential equations. The most popular approach was reported by Monaghan [116], who used Equation (3.22) and Equations (3.19) directly. Monaghan suggested to put the density within the gradient operator:

$$\nabla \cdot A = \frac{1}{\rho}[\nabla \cdot (\rho A) - A \cdot \nabla \rho], \tag{3.24}$$

or

$$\nabla \cdot A = \rho \left[\nabla \cdot \left(\frac{A}{\rho} \right) + \frac{A}{\rho^2} \cdot \nabla \rho \right] \tag{3.25}$$

Now, Equation (3.24) is rewritten as follows:

$$\begin{aligned}
(\nabla A)_i &= \frac{1}{\rho_i}[\nabla \cdot (\rho A) - A \cdot \nabla \rho]_i \\
&\cong \frac{1}{\rho_i}\nabla \cdot \int_\Omega (\rho A) W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}' - \frac{A}{\rho_i} \cdot \int_\Omega (\nabla \rho) W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}' \\
&\cong \frac{1}{\rho_i} \sum_{j=1}^{N} \frac{m_j}{\rho_j} (\rho A)_j \cdot \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) - \frac{A_j}{\rho_i} \cdot \sum_{j=1}^{N} \frac{m_j}{\rho_j} \rho_j \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \\
&= \frac{1}{\rho_i} \sum_{j=1}^{N} m_j A_j \cdot \nabla W_{ij} - \frac{1}{\rho_i} \sum_{j=1}^{N} m_j A_i \cdot \nabla W_{ij} \\
&= -\frac{1}{\rho_i} \sum_{j=1}^{N} m_j A_{ij} \cdot \nabla_i W_{ij}
\end{aligned} \tag{3.26}$$

where $A_{ij} = A_i - A_j$ and $W_{ij} = W(\mathbf{r}_i - \mathbf{r}_j, h)$.

Similarly, for Equation (3.25):

$$\begin{aligned}
(\nabla A)_i &= \rho_i \left[\nabla \cdot \left(\frac{A}{\rho} \right) + \frac{A}{\rho^2} \cdot \nabla \rho \right]_i \\
&\cong \rho_i \nabla \cdot \int_\Omega \left(\frac{A}{\rho} \right) W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}' + \rho_i \cdot \frac{A}{\rho_i^2} \cdot \int_\Omega (\nabla \rho) W(\mathbf{r} - \mathbf{r}', h) d\mathbf{r}' \\
&\cong \rho_i \sum_{j=1}^{N} \frac{m_j}{\rho_j} \left(\frac{A}{\rho} \right)_j \cdot \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) + \frac{A_j}{\rho_j} \cdot \sum_{j=1}^{N} \frac{m_j}{\rho_j} \rho_j \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \\
&= \rho_i \left(\sum_{j=1}^{N} \frac{m_j A_j}{\rho_j^2} \cdot \nabla W_{ij} + \frac{A_i}{\rho_i^2} \sum_{j=1}^{N} m_j \cdot \nabla W_{ij} \right) \\
&= \rho_i \sum_{j=1}^{N} m_j \left(\frac{A_j}{\rho_j^2} + \frac{A_i}{\rho_i^2} \right) \cdot \nabla_i W_{ij}
\end{aligned} \tag{3.27}$$

Equations (3.26) and (3.27) are symmetric and antisymmetric, respectively, when i

and j are exchanged. Equation (3.27) is usually used for pressure gradient as it help conserves of momentum [116]. Another form was suggested by Bonet [28], and it is given in the form as following:

$$(\nabla A)_i = \sum_{j=1}^{N} \frac{m_j}{\rho_j}(A_j + A_i) \cdot \nabla_i W_{ij}. \tag{3.28}$$

3.5 Kernel Approximation

The SPH kernel must satisfy the three condition given by Equations (3.5), (3.6) and (3.8). Different kinds of kernel functions exist and some of the popular ones along with their characteristics are presented in the following subsections.

Bell-shaped kernel function

Lucy in his paper [97] used bell-shaped kernel (Figure 3.4) defined by:

$$W(\mathbf{r} - \mathbf{r}', h) = W(q, h) = \alpha_d \begin{cases} (1+3q)(1-q)^3 & q \leq 1 \\ 0 & q > 1, \end{cases} \tag{3.29}$$

$$\frac{dW}{dr}(q, h) = \frac{\alpha_d}{h} \begin{cases} -12q(1-q)^2 & q \leq 1 \\ 0 & q > 1, \end{cases} \tag{3.30}$$

where $\alpha_d = \begin{cases} 5/(4h) & \text{in one dimensional space} \\ 5/(\pi h^2) & \text{in two dimensional space} \\ 105/(16\pi h^3) & \text{in three dimensional space} \end{cases}$, for the unity requirement

and $q = \dfrac{|\mathbf{r} - \mathbf{r}'|}{h}$ is the relative distance between $\mathbf{r}$ and $\mathbf{r}'$

The supporting domain of this kernel function is small which leads to a decrease in number of neighbour particles and thus, affects the accuracy of the problem.

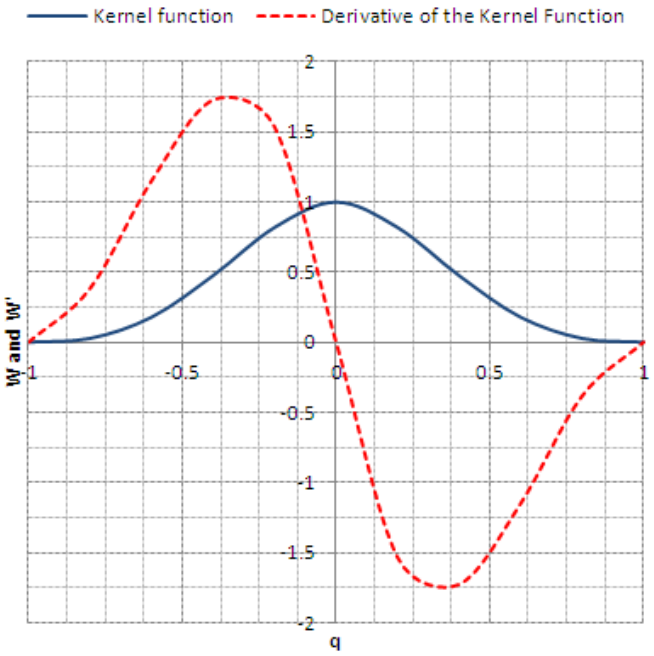


FIGURE 3.4: The kernel function used by Lucy [97]. The supporting domain of this kernel function is small and this leads to decrease in number of neighbour particles and effects to the accuracy of the problem.

Gaussian kernel function

According to Monaghan [116], Gaussian kernel was one of the best function to simulate non-spherical stars (Figure 3.5) where the following kernel was used:

$$W(q,h) = \alpha_d e^{-q^2}, \tag{3.31}$$

$$\frac{dW}{dr}(q,h) = \frac{-2q\alpha_d}{h} e^{-q^2}, \tag{3.32}$$

$$\text{where } \alpha_d = \begin{cases} 1/(\pi^{1/2}h) & \text{in one dimensional space} \\ 1/(\pi h^2) & \text{in two dimensional space} \\ 1/(\pi^{3/2}h^3) & \text{in three dimensional space} \end{cases}$$

The Gaussian kernel is smooth and very stable but not compact. However, this function never tends to zero except when q becomes infinity. It means that, the Gaussian kernel function is computationally more expensive.

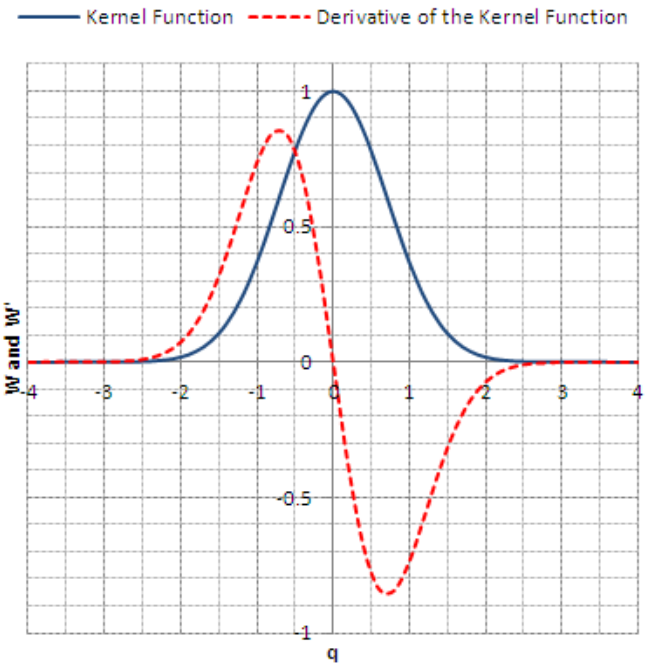


FIGURE 3.5: The Gaussian kernel function is computationally more expensive since it takes a longer path for the function to approach zero.

B-spline kernel function

The B-spline kernel function based on the cubic spline and was developed by Monaghan and Lattanzio [119] (Figure 3.6). The 2nd derivative of this function is partly a linear function.

$$W(q,h) = \alpha_d \begin{cases} \frac{2}{3} - q^2 + \frac{1}{2}q^3 & 0 \leq q < 1 \\ \frac{1}{6}(2-q)^3 & 1 \leq q < 2 \\ 0 & q \geq 2, \end{cases} \tag{3.33}$$

$$\frac{dW}{dr}(q,h) = \frac{\alpha_d}{h} \begin{cases} -2q + \frac{3}{2}q^2 & 0 \leq q < 1 \\ -\frac{1}{2}(2-q)^2 & 1 \leq q < 2 \\ 0 & q \geq 2, \end{cases} \tag{3.34}$$

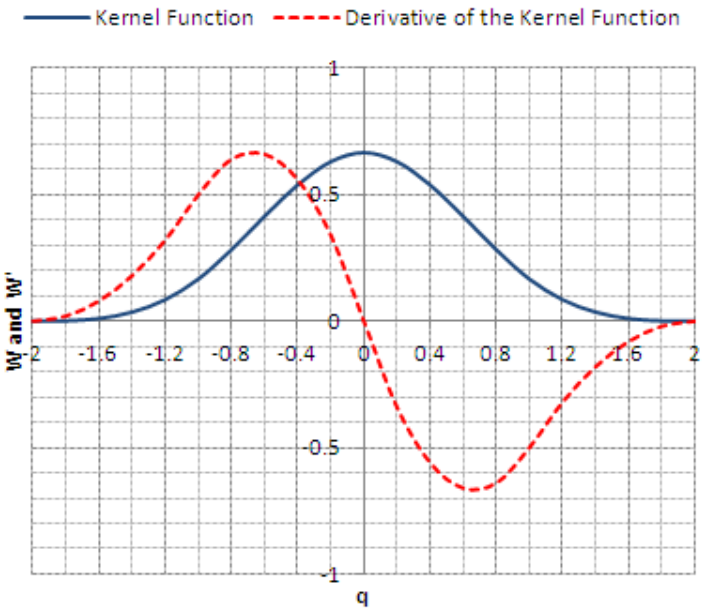


FIGURE 3.6: The B-spline kernel function with supporting domain of $2h$ and ensures a net limit on the number of neighbour particles.

where

$$\alpha_d = \begin{cases} 1/h & \text{in one dimensional space} \\ 15/(7\pi h^2) & \text{in two dimensional space} \\ 3/(2\pi h^3) & \text{in three dimensional space} \end{cases}$$

The kernel tends to zero at a distance $2h$ from its peak which means it has a compact support. This ensures a net limit on the number of neighbour particles which ensures reasonable accuracy.

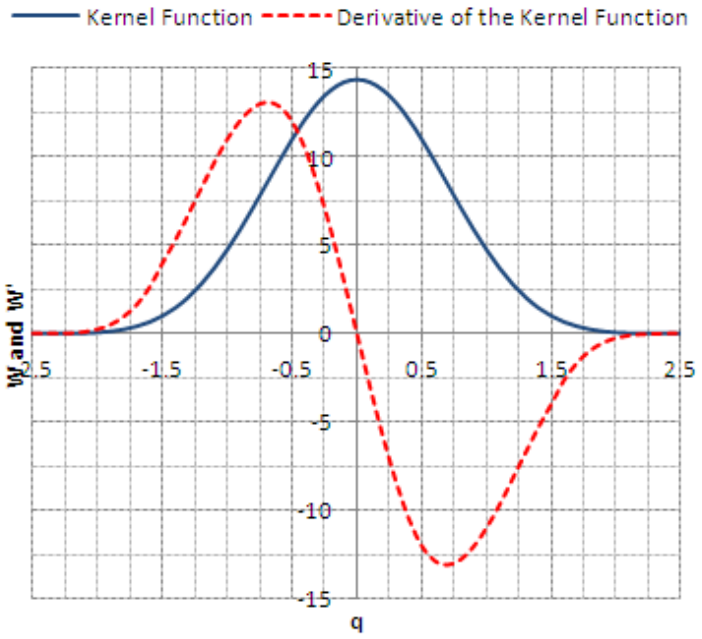


FIGURE 3.7: The quartic kernel function with a compact supporting length of $2.5h$.

Quartic and quintic kernel functions

Higher order splines (quartic and quintic) introduced by Morris [120, 121], are more stable. The quartic function is given by (see Figure 3.7):

$$W(q,h) = \alpha_d \begin{cases} (q+2.5)^4 - 5(q+1.5)^4 + 10(q+0.5)^4 & 0 \le q < 0.5 \\ (2.5-q)^4 - 5(1.5-q)^4 & 0.5 \le q < 1.5 \\ (2.5-q)^4 & 1.5 \le q < 2.5 \\ 0 & q > 2.5, \end{cases} \tag{3.35}$$

$$\frac{dW}{dr}(q,h) = \frac{\alpha_d}{h} \begin{cases} 4(q+2.5)^3 - 20(q+1.5)^3 + 40(q+0.5)^3 & 0 \le q < 0.5 \\ -4(2.5-q)^3 + 20(1.5-q)^3 & 0.5 \le q < 1.5 \\ -4(2.5-q)^3 & 1.5 \le q < 2.5 \\ 0 & q > 2.5, \end{cases} \tag{3.36}$$

where $\alpha_d = 1/24h$ in one dimension. The quintic function is given by (see Figure 3.8):

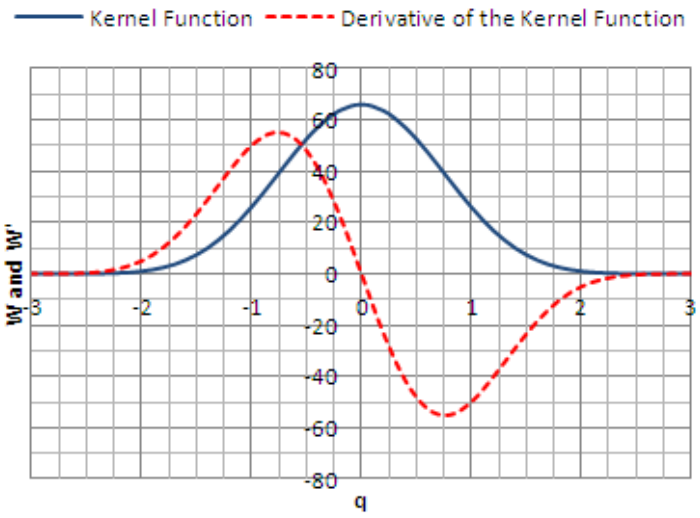


FIGURE 3.8: The quintic kernel function with a compact supporting length of $3h$ and have more stable properties with increased computational effort.

$$W(q,h) = \alpha_d \begin{cases} (3-q)^5 - 6(2-q)^5 + 15(1-q)^5 & 0 \le q < 1 \\ (3-q)^5 - 6(2-q)^5 & 1 \le q < 2 \\ (3-q)^5 & 2 \le q < 3 \\ 0 & q > 3, \end{cases} \tag{3.37}$$

$$\frac{dW}{dr}(q,h) = \frac{\alpha_d}{h} \begin{cases} -5(3-q)^4 + 30(2-q)^4 - 75(1-q)^4 & 0 \le q < 1 \\ -5(3-q)^4 + 30(2-q)^4 & 1 \le q < 2 \\ -5(3-q)^4 & 2 \le q < 3 \\ 0 & q > 3, \end{cases} \tag{3.38}$$

$$\text{where } \alpha_d = \begin{cases} 120/h & \text{in one dimensional space} \\ 7/(478\pi h^2) & \text{in two dimensional space} \\ 3/(359\pi h^3) & \text{in three dimensional space} \end{cases}$$

Quartic and quintic kernels have gradually better stability properties but at an increased computational effort because the region of contributing neighbours is larger.

Wendland kernel functions

Dehnen and Aly [43] concluded in their joint work that Wendland [155] kernel functions are ideal candidate for SPH smoothing kernel in order to avoid the pairing instability for free surface problems.

$$W(\mathbf{r}-\mathbf{r}',h) = W(q,h) = \alpha_d \begin{cases} \left(1-\frac{q}{2}\right)^4 (1+2q) & 0 \le q \le 2 \\ 0 & q > 2, \end{cases} \tag{3.39}$$

$$\frac{dW}{dr}(q,h) = \frac{\alpha_d}{h} \begin{cases} -5q\left(1-\frac{q}{2}\right)^3 & 0 \le q \le 2 \\ 0 & q > 2, \end{cases} \tag{3.40}$$

where $\alpha_d = \dfrac{7}{4\pi h^2}$ for two dimensional space

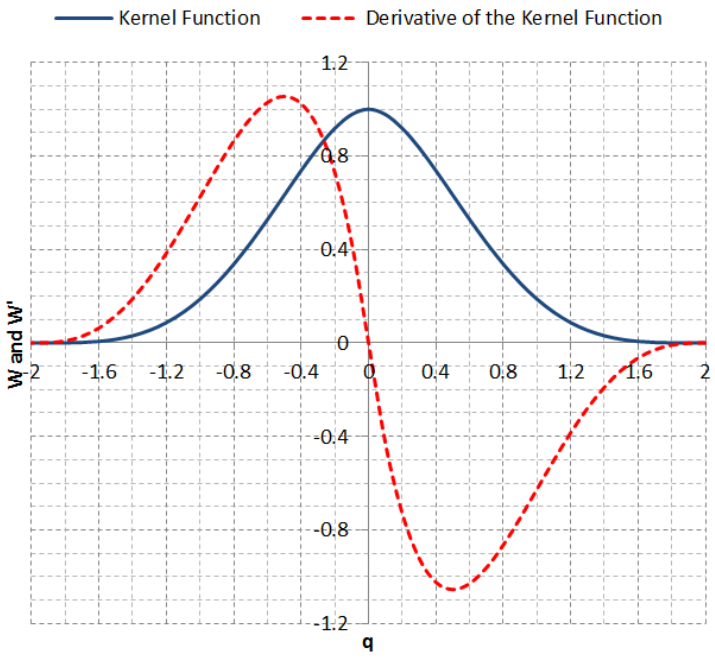


FIGURE 3.9: The Wendland [155] kernel function and its first derivative.

3.6 SPH discretisation for Navier-Stokes equations

Navier-Stokes formulation for conservation of mass, conservation of momentum and conservation of energy in the Lagrangian form are given as follows:

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot\mathbf{v}, \tag{3.41}$$

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \frac{1}{\rho}\nabla\cdot\tau + \frac{\mathbf{F}}{\rho}, \tag{3.42}$$

$$\frac{De}{Dt} = -\frac{p}{\rho}\nabla\cdot\mathbf{v} + \frac{1}{\rho}\tau : \nabla\mathbf{v} - \frac{1}{\rho}\nabla\cdot\mathbf{q}. \tag{3.43}$$

where p is the pressure, $\mathbf{v}$ is the particle velocity, t is the time, ρ is the density, $\mathbf{F}$ is a body force, $\mathbf{q}$ is the heat flux and τ is the viscous stress tensor given by:

$$\tau = 2\mu\varepsilon - \frac{2}{3}\mu(\nabla\cdot\mathbf{v})I. \tag{3.44}$$

where μ is the viscosity, I is unit tensor, ε is a change of deformation tensor:

$$\varepsilon = \frac{1}{2}\left[\nabla\cdot\mathbf{v} + (\nabla\cdot\mathbf{v})^T\right]. \tag{3.45}$$

3.6.1 Particle approximation for density

The density approximation in SPH is very important because it influences its neighbouring particle distribution. There are two approaches to calculate density in SPH. The first one is known as the summation density approach where for a given particle i, the density is calculated from the Equation (3.19) by directly replacing $A(\mathbf{r})$ with density, ρ:

$$\rho_i = \sum_{j=1}^{N} \frac{m_j}{\rho_j}\rho_j \cdot W_{ij}, \tag{3.46}$$

$$\rho_i = \sum_{j=1}^{N} m_j W_{ij} \tag{3.47}$$

where W_{ij} has a unit of the inverse of volume.

Density truncates notably close to the edge of the fluid with Equation (3.47), and known as deficiency of the boundary particles. According to Randles and Libersky[132], normalising the summation density improves the accuracy of the summation density equation:

$$\rho_i = \frac{\sum_{j=1}^{N} m_j W_{ij}}{\sum_{j=1}^{N} \frac{m_j}{\rho_j} W_{ij}} \tag{3.48}$$

The second approach is the continuity density; Here, different operations on the right hand side of the continuity Equation (3.41) will yield different forms for density approximation. By applying SPH discretisation (see Equation (3.26)) to the velocity divergence in Equation (3.41) yields:

$$\nabla \cdot \mathbf{v}_i \approx -\frac{1}{\rho_i} \sum_{j=1}^{N} m_j \mathbf{v}_{ij} \cdot \nabla_i W_{ij} \tag{3.49}$$

where $\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j$

Substituting the gradient of velocity into Equation (3.41) produces:

$$\frac{D\rho}{Dt} \approx \sum_{j}^{N} m_j \mathbf{v}_{ij} \cdot \nabla_i W_{ij} \tag{3.50}$$

Thus, the density with equation (3.50) changes when particles move with different velocities relative to one another.

Following Mahdavi and Talebbeydokhti [99], the above equation can be stabilized with the method proposed by Ferrari *et al.* [52]. This stabilized density equation is simple to implement and there is no need to calibrate the parameters. As it was mentioned in Mahdavi and Talebbeydokhti [99], this approach helps to remove spurious oscillations in the flow field. Therefore, the continuity density Equation (3.50) takes the following form:

$$\frac{D\rho}{Dt} = \sum_{j}^{N} m_j \mathbf{v}_{ij} \cdot \nabla_i W_{ij} + \sum_{j}^{N} c_{ij} \frac{m_j}{\rho_j} (\rho_j - \rho_i) \mathbf{n}_{ij} \cdot \nabla_i W_{ij} \tag{3.51}$$

where $\mathbf{n}_{ij} = \mathbf{r}_{ji}/|\mathbf{r}_{ji}|$ and $c_{ij} = \max(c_i, c_j)$ corresponds to the greatest celerity among the two neighbouring interacting particles.

$$c_i = c_s \sqrt{\left(\frac{\rho_i}{\rho_0}\right)^{\gamma-1}} \tag{3.52}$$

where c_s is the speed of sound, $\gamma = 7$ and ρ_0 is the reference density. This density approach is only used for dam break problem that can be seen later in chapter 4

3.6.2 Particle approximation for momentum

From the standard formulation of the SPH (Equation (3.26)), the discretised form of the acceleration due to the pressure gradient can be written as:

$$\begin{aligned}\left(\frac{D\mathbf{v}}{Dt}\right)_p &= \left(-\frac{1}{\rho}\nabla p\right)_i \\ &= -\frac{1}{\rho_i}\sum_{j=1}^{N}\frac{m_j}{\rho_j}(p_j - p_i)\cdot\nabla_i W_{ij}.\end{aligned} \tag{3.53}$$

It is very important to ensure that angular and linear momentum conserve exactly due to the discretised form of the forces between neighbour particles during the discretisation of the terms in the momentum equation. The angular and linear momentum do not conserve in above equation. Monaghan [117] suggested another way of discretisation which ensures exact conservation of momentum. The derivation of SPH formulations for particle approximation of momentum is somewhat similar to the continuity density approach in Monaghan's approach (Equation (3.27)). Applying the SPH particle approximation methods to the pressure gradient on the right hand side of the momentum Equation (3.42) using the discretised expression in Equation (3.27)

is rewritten as:

$$\begin{aligned}\left(\frac{D\mathbf{v}}{Dt}\right)_p &= \left(-\frac{1}{\rho}\nabla p\right)_i \\ &= -\frac{1}{\rho_i}\cdot\rho_i\sum_{j=1}^{N} m_j\left(\frac{p_j}{\rho_j^2}+\frac{p_i}{\rho_i^2}\right)\cdot\nabla_i W_{ij} \\ &= -\sum_{j=1}^{N} m_j\left(\frac{p_j}{\rho_j^2}+\frac{p_i}{\rho_i^2}\right)\cdot\nabla_i W_{ij}. \end{aligned} \tag{3.54}$$

or it can be also written in the form of Bonet's [28] suggestion as in Equation (3.28)

$$\begin{aligned}\left(\frac{D\mathbf{v}}{Dt}\right)_p &= \left(-\frac{1}{\rho}\nabla p\right)_i \\ &= -\frac{1}{\rho_i}\cdot\rho_i\sum_{j=1}^{N} \frac{m_j}{\rho_j}(p_j+p_i)\cdot\nabla_i W_{ij} \\ &= -\sum_{j=1}^{N} m_j\left(\frac{p_j+p_i}{\rho_j\rho_i}\right)\cdot\nabla_i W_{ij}. \end{aligned} \tag{3.55}$$

In a similar way, applying the SPH particle approximation methods to the viscous stress tensor term on the right hand side of the momentum Equation (3.42) gives:

$$\begin{aligned}\left(\frac{D\mathbf{v}}{Dt}\right)_\tau &= \left(\frac{1}{\rho}\nabla\cdot\tau\right)_i \\ &= \frac{1}{\rho_i}\cdot\rho_i\sum_{j=1}^{N} m_j\left(\frac{\tau_j}{\rho_j^2}+\frac{\tau_i}{\rho_i^2}\right)\cdot\nabla_i W_{ij} \\ &= \sum_{j=1}^{N} m_j\left(\frac{\tau_j}{\rho_j^2}+\frac{\tau_i}{\rho_i^2}\right)\cdot\nabla_i W_{ij}. \end{aligned} \tag{3.56}$$

Finally, the particle approximation of momentum equation is written as

$$\frac{D\mathbf{v}}{Dt} = -\sum_{j=1}^{N} m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2}\right) \cdot \nabla_i W_{ij} + \sum_{j=1}^{N} m_j \left(\frac{\tau_j}{\rho_j^2} + \frac{\tau_i}{\rho_i^2}\right) \cdot \nabla_i W_{ij} + \frac{\mathbf{F}}{\rho_i}. \tag{3.57}$$

The particle approximation of the viscous stress tensor term τ is expressed as following [104]:

$$\tau_i = -\sum_{j=1}^{N} \frac{m_j}{\rho_j} \mu_i \mathbf{v_{ij}} \nabla_i W_{ij} - \sum_{j=1}^{N} \frac{m_j}{\rho_j} \mu_i (\nabla_i W_{ij}) \mathbf{v_{ij}} + \left(\frac{2}{3} \sum_{j=1}^{N} \frac{m_j}{\rho_j} \mu_i \mathbf{v_{ij}} \cdot \nabla_i W_{ij}\right) I. \tag{3.58}$$

For the Newtonian fluid such as water and air, the momentum equation can be rewritten as follows:

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \cdot \mathbf{v} + \frac{\mathbf{F}}{\rho}. \tag{3.59}$$

Monaghan [118] used the following approach for approximating of the viscous term in the momentum equation for Newtonian fluid, which is more suitable for modelling low velocity fluid flows. In Monaghan's approach, the momentum equation for viscous term is as follows:

$$\begin{aligned} \left(\frac{D\mathbf{v}}{Dt}\right)_{\mu} &= \left(\frac{\mu}{\rho}\nabla^2 \cdot \mathbf{v}\right)_i \\ &= \sum_{j=1}^{N} m_j \left(\frac{\mu_i + \mu_j}{\rho_i \rho_j}\right) \mathbf{v}_{ij} \left(\frac{1}{r_{ij}} \frac{dW_{ij}}{dr_{ij}}\right). \end{aligned} \tag{3.60}$$

The momentum equation, that we use for our simulation is a combination of Equations (3.54) and (3.60):

$$\frac{D\mathbf{v}}{Dt} = -\sum_{j=1}^{N} m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2}\right) \cdot \nabla_i W_{ij} + \sum_{j=1}^{N} m_j \left(\frac{\mu_i + \mu_j}{\rho_i \rho_j}\right) \mathbf{v}_{ij} \left(\frac{1}{r_{ij}} \frac{dW_{ij}}{dr_{ij}}\right) + \frac{\mathbf{F}}{\rho_i}. \tag{3.61}$$

3.6.3 Particle approximation for energy

The first part of the internal energy, e, in Equation (3.43) involving pressure work can be approximated according to the Equation (3.26):

$$\left(-\frac{p}{\rho}\nabla\cdot\mathbf{v}\right)_i = \frac{p_i}{\rho_i^2}\sum_{j=1}^{N} m_j \mathbf{v_{ij}}\cdot\nabla_i W_{ij}. \tag{3.62}$$

Also it can be written according to Equation (3.25) such that:

$$-\frac{p}{\rho}\nabla\cdot\mathbf{v} = -\nabla\cdot\left(\frac{p}{\rho}\mathbf{v}\right) + \mathbf{v}\cdot\left(\frac{p}{\rho}\right). \tag{3.63}$$

The SPH formulation for the above equation is similar to the Equation (3.27) and can be expressed as:

$$\left(-\frac{p}{\rho}\nabla\cdot\mathbf{v}\right)_i = \sum_{j=1}^{N} m_j \frac{p_j}{\rho_j^2}\mathbf{v_{ij}}\cdot\nabla_i W_{ij}. \tag{3.64}$$

To obtain the symmetric SPH formulation for the internal energy equation due to compression is to take the average of Equations (3.62) and (3.64):

$$\left(-\frac{p}{\rho}\nabla\cdot\mathbf{v}\right)_i = \frac{1}{2}\sum_{j=1}^{N} m_j\left(\frac{p_i}{\rho_i^2}+\frac{p_j}{\rho_j^2}\right)\mathbf{v_{ij}}\cdot\nabla_i W_{ij}. \tag{3.65}$$

From the standard formulation of the SPH, the discretised form of the internal energy due to the viscous dissipation is written as:

$$\left(\frac{1}{\rho}\tau:\nabla\mathbf{v}\right)_i = -\sum_{j=1}^{N} m_j \frac{\tau_i}{\rho_i^2}:\mathbf{v_{ij}}\nabla_i W_{ij}. \tag{3.66}$$

And also it can be expressed in another form according to Equation (3.25):

$$\begin{aligned}\left(\frac{1}{\rho}\tau:\nabla\mathbf{v}\right)_i &= \nabla\cdot\left(\frac{\tau_i}{\rho_i}\cdot\mathbf{v_i}\right) - \mathbf{v_i}\cdot\left(\nabla\cdot\frac{\tau_i}{\rho_i}\right)\\ &= -\sum_{j=1}^{N} m_j\frac{\tau_j}{\rho_j^2}:\mathbf{v_{ij}}\nabla_i W_{ij}.\end{aligned} \tag{3.67}$$

Finally, the SPH formulation for the viscous dissipation term of the internal energy is:

$$\left(\frac{1}{\rho}\tau : \nabla \mathbf{v}\right)_i = -\frac{1}{2}\sum_{j=1}^{N} m_j \left(\frac{\tau_i}{\rho_i^2} + \frac{\tau_j}{\rho_j^2}\right) : \mathbf{v_{ij}} \nabla_i W_{ij}. \tag{3.68}$$

The conduction term of the internal energy can be expressed as:

$$\begin{aligned}\left(-\frac{1}{\rho}\nabla \cdot \mathbf{q}\right)_i &= -\nabla \cdot \left(\frac{\mathbf{q_i}}{\rho_i}\right) - \frac{\mathbf{q_i}}{\rho_i^2} \cdot \nabla \rho_i \\ &= -\sum_{j=1}^{N} m_j \left(\frac{\mathbf{q_j}}{\rho_j^2}\right) \cdot \nabla_i W_{ij} - \frac{\mathbf{q_i}}{\rho_i^2} \cdot \sum_{j=1}^{N} m_j \nabla_i W_{ij} \\ &= -\sum_{j=1}^{N} m_j \left(\frac{\mathbf{q_j}}{\rho_j^2} + \frac{\mathbf{q_i}}{\rho_i^2}\right) \cdot \nabla_i W_{ij}. \end{aligned} \tag{3.69}$$

The conduction term is based on the Fourier's law, and is discretised as follows:

$$\mathbf{q}_i = -k_i \nabla T_i = \frac{1}{\rho_i}\sum_{j=1}^{N} m_j T_{ij} \nabla_i W_{ij}. \tag{3.70}$$

Here $T_{ij} = T_i - T_j$ is the difference in temperature. The final discretised form for the SPH formulation for the internal energy equation is:

$$\begin{aligned}\frac{De_i}{Dt} = \frac{1}{2}\sum_{j=1}^{N} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right) \mathbf{v_{ij}} \cdot \nabla_i W_{ij} - \frac{1}{2}\sum_{j=1}^{N} m_j \left(\frac{\tau_i}{\rho_i^2} + \frac{\tau_j}{\rho_j^2}\right) : \mathbf{v_{ij}} \nabla_i W_{ij} \\ - \sum_{j=1}^{N} m_j \left(\frac{\mathbf{q_j}}{\rho_j^2} + \frac{\mathbf{q_i}}{\rho_i^2}\right) \cdot \nabla_i W_{ij}. \end{aligned} \tag{3.71}$$

3.7 Equation of state

An equation relating the pressure and the density is required to close the N-S Equations (3.61). This is known as an equation of state and can comprise either a differential or an algebraic formulation. An equation of state can be solved in 2 possible ways namely weakly compressible SPH (WSPH) and incompressible SPH (ISPH).

1. Monaghan [117] and Batchelor [11] suggested using Tait's equation as the equation of state in the WSPH, where the pressure depends on the density and expressed as

$$p = B\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] \tag{3.72}$$

where γ is a polytrophic constant which varies between 1 and 7. For particular case which is water $\gamma = 7$ [107], ρ_0 is the reference density and B is the reference pressure constant and is written as

$$B = \frac{\rho_0 c_s^2}{\gamma} \tag{3.73}$$

where c_s is the speed of sound which controls the compressibility of the fluid. Monaghan [117] showed that we can choose the sound speed to be one hundred times greater than the maximum speed of the fluid flow in order to reduce the fluctuation in density to within 1%.

As it can be seen from the above equation, that a small change in density leads to a large oscillation in pressure. The subtraction of 1 in the equation of state is to ensure zero pressure at a free surface flow.

2. The next option is to use a differential equation that ensure ISPH. This involves solving the pressure Poisson equation [86].

$$\nabla \cdot \left(-\frac{1}{\rho}\nabla p\right)_i = \frac{1}{\Delta t}\nabla \cdot \mathbf{v}_i^* \tag{3.74}$$

where $\mathbf{v}^*$ is an intermediate velocity calculated by taking the solution over one time-step while ignoring the pressure. Solving the Poisson's equation needs different methodology, like the projection approach [38] which is very expensive computationally. Furthermore, there are also issues of free-surface instability that has been reported [78, 86]. This need to be corrected which will also lead to more computational cost.

Due to the above arguments, the Tait's equation is chosen for our WSPH simulation based on the fact that it solve much quicker than the Poisson's equation despite the large pressure fluctuations which can be controlled by varying the sound speed.

3.8 The Smoothing Length

The smoothing length is a key parameter in the SPH method. It defines the distance within which particles act with each other given by:

$$R = k \cdot h. \tag{3.75}$$

The k is determined by the choice of the smoothing kernel. If $k = 2$, particles at a distance greater than two smoothing lengths will have no influence on the particle. In other word, the smoothing kernel is zero when the distance to the neighbour particle is greater than $2h$.

3.9 Neighbouring Particles Search

One of the important steps in SPH method is determining a particles neighbours that lies within its support domain. For a given particle, the number of neighbouring particles can vary with time as particles collide or move from one place to another.

3.9.1 All-pair Search Method

The simplest method to determining neighbouring particles is the all-pair search method. This method does not search for influencing pairs of particles, but checks all particles pairs for interactions; The search is carried out for all N particles in the problem domain and then chose those neighbour particles within the supporting domain (kh) as shown in Figure 3.10. The all-pair search method has a complexity of order

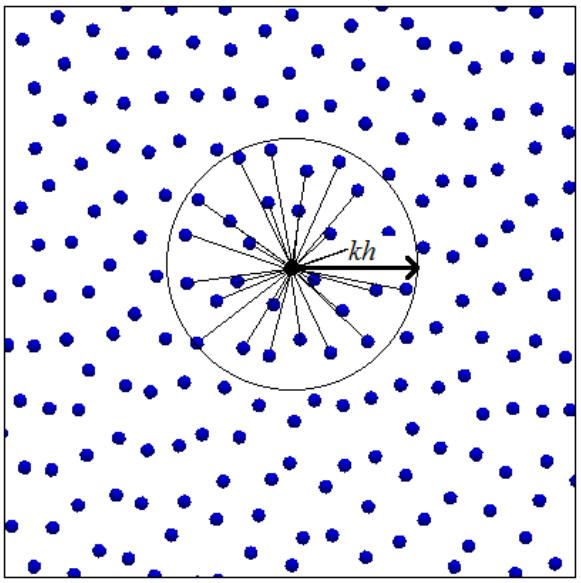


FIGURE 3.10: The all-pair method in two-dimensional space checks all pairs of particles for interactions and the searching is carried out for all the particles N and only then chose neighbour particles within the supporting domain (kh).

$O(N^2)$. As the search process is necessary for all the time steps, the computational time will be expensive.

3.9.2 Cell Search Method

The cell search method is widely used in SPH simulations because the search is carried out only for a certain group of particles while in all-pair search method, the search is carried out for entire domain, and works well for cases with constant smoothing length. If all the particles are located into cells then searching for nearest neighbours is only needed for a small group of particles. For example, for a particle i, particles located in the neighbouring or same cells are tested for possible interaction (Figure 3.11). The number of search cells is 3, 9, and 27 cells for one, two, and three-dimensional spaces, respectively. The complexity of the cell search is of order $O(N)$ in case of the number of particles per cell is reasonably small.

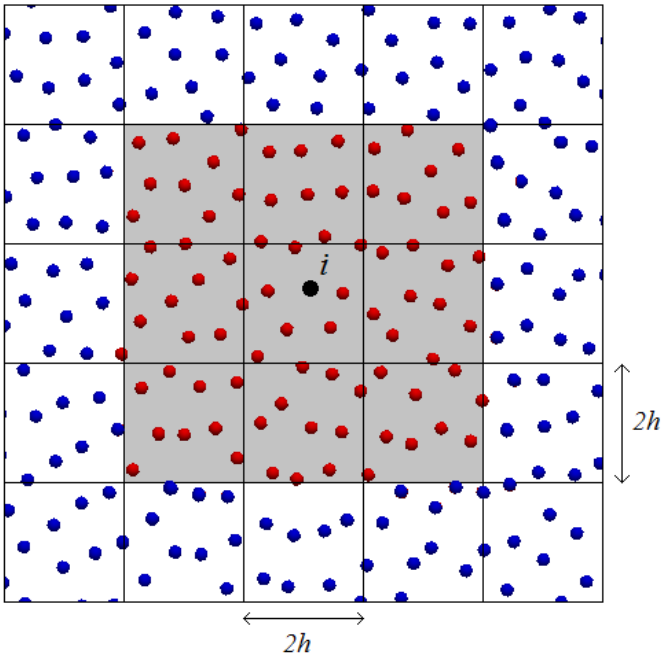


FIGURE 3.11: The cell search method in two-dimensional space for a given particle i (black point particle), particles located in the neighbouring or same cells (shaded cells) are tested for possible interaction.

3.9.3 Kd-Tree Method

Kd-tree is generalized form of the standard one-dimensional binary tree in which every node is a k - dimensional space, developed by Bentley [15]. The root node (parent) describes the whole space, and the leaf nodes (children) describe subspaces containing relatively small subsets of data. Only one of the k - dimensions is used as a key to divide the space at any node and this division continues until the smallest subspace contains only one particle. Division of the entire space in two subspaces occure in the order according to its dimensions. For example, the entire space is divided first by x component, next division of the subspaces in left and right side by y component and so on. It means that all particles on the left side clearly is less in x direction than the those on the right side. When the tree searching is started, the value of the key is compared with the node key value, and the corresponding branch is followed. When a leaf node is reached, all of the particles resident in the leaf's area are tested, and

the particles in the support domain are determined [65]. The efficiency of the kd-tree search is $O(\log N)$.

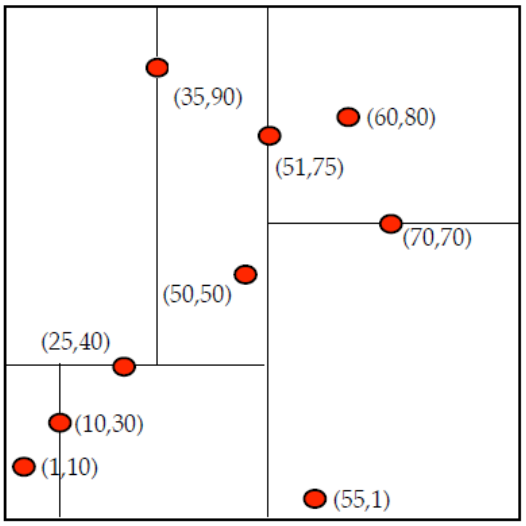


FIGURE 3.12: The KD-Tree method.

3.10 Boundary Conditions

3.10.1 Solid Walls

In general, in fluid flow modelling, the need to model solid walls or boundaries as well as fluid particles. These solid walls can be modelled in different ways. Here, in this section, only the most used and popular ones in SPH will be explained.

The wall is often treated as SPH particles located at a mainly constant interval (usually this interval is equal to the diameter of the particle). In the SPH literatures [93, 151], it is called as wall particles or edge particles (dynamic boundary) and these particles are treated like normal fluid particles without updating their initial velocity and position as illustrated in Figure 3.13 below.

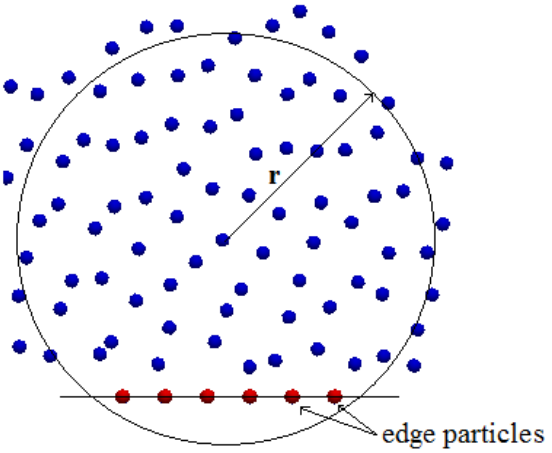


FIGURE 3.13: Edge particles fixed in one layer and located at a mainly constant interval.

When the fluid particles approach near to the edge, as shown in Figure 3.14, the number of the neighbor particles significantly decreases within its supporting domain. For this reason, forces may not balanced due to the truncation of neighbor particles and the fluid particle may leave the fluid domain.

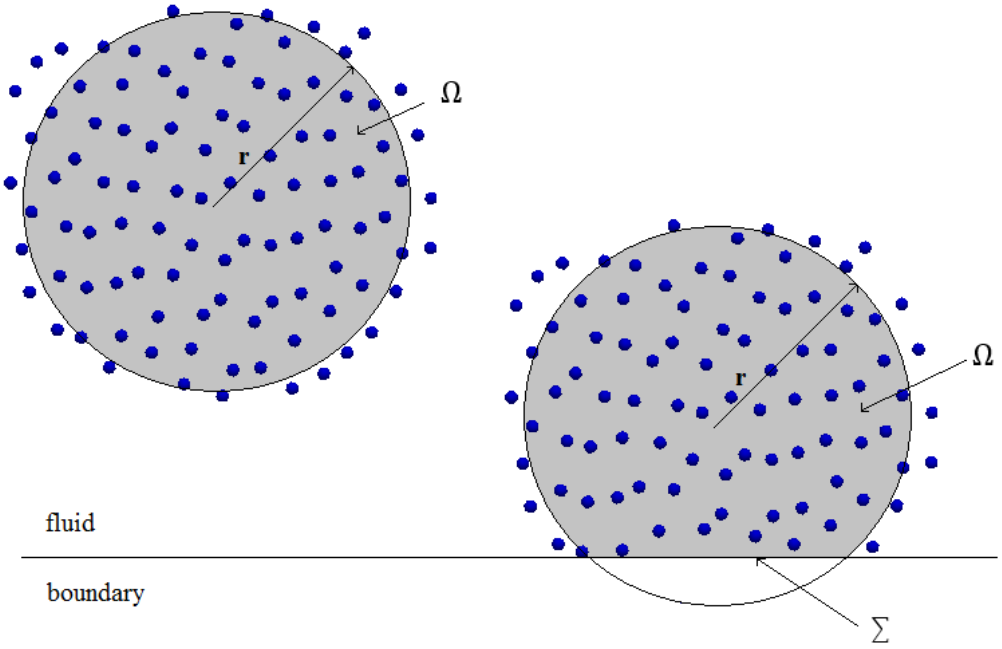


FIGURE 3.14: A void in a supporting domain of the fluid particles near the edge boundary.

To avoid these disadvantages of the above methods, Monaghan [117] suggested different way in which we do not need any extra particle beyond the edge particles. To keep the fluid particles within the fluid domain, the extra repulsive force was introduced between the edge particles and the fluid particles. And this extra repulsive force is included to the momentum Equation (3.61). The most often used extra repulsive force is based on the Lennard-Jones molecular model. This force per unit mass in the Lennard-Jones form is given as following equation

$$f_i(\mathbf{r}) = gH \left[\left(\frac{r_0}{r_{ij}} \right)^{P_2} - \left(\frac{r_0}{r_{ij}} \right)^{P_1} \right] \frac{\mathbf{r_{ij}}}{r_{ij}^2} \tag{3.76}$$

where $P_2 = 4$ and $P_1 = 2$, r_0 is a initial separation of the particles, g is the acceleration of gravity and H is a height of the fluid.

3.10.2 The dummy particles

The above mentioned issue can be solved in different ways. One of the ways is by introducing dummy particles on the other side of the edge particles. Basically, this is a way of avoiding truncation of neighbour particles in the support domain for fluid particles near the edge particles. The dummy particles are stationary particles that

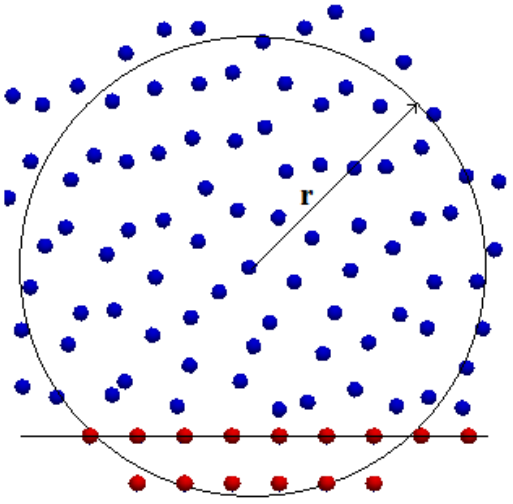


FIGURE 3.15: The dummy particles are introduced to avoid the truncation of neighbour particles in supporting domain.

are spaced equally from each other with the same initial separation of fluid particles in a few layers behind the edge particles (see Figure 3.15). Logically, the number of the layers depends on the type of the kernel function, but generally, this number is not exceed two or three layers. The role of these dummy particles is just to exist there where they are located initially and to keep all initial physical quantities constant over the time and to provide a full kernel for near edge fluid particles.

3.10.3 The mirror particles

Another and also popular method in the way of introducing the wall boundaries is the mirror particle approach. The mirror particle method is similar to the dummy particle method as was described above, but in contrast to the dummy particle, the physical quantities of the mirror particles depend to the fluid particles near the wall. For each fluid particles near the wall or edge particles, the mirror particles are created with symmetric parameters to the fluid particles (see Figure 3.16). A non-slip condition can be handled without difficulty by introducing the same velocity of the fluid particles to their mirror particles with opposite direction[151].

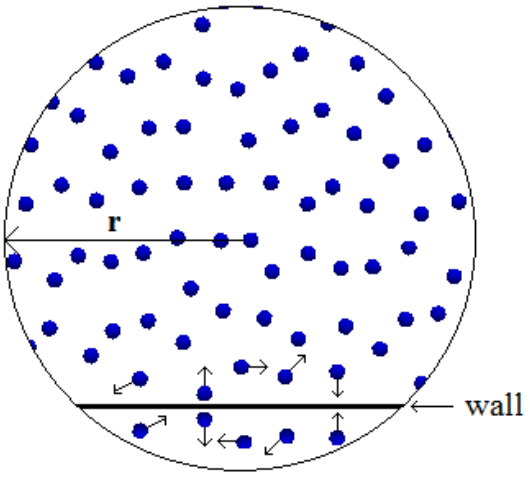


FIGURE 3.16: The mirror particles are introduced with symmetric parameters of the corresponding fluid particles. The arrows show the directions of movement.

The dummy particle and the mirror particle methods are simple in terms of application, but in terms of the computational cost, they might be very expensive for some

cases, especially for three-dimensional problems and takes more memory space.

3.10.4 The periodic boundary

For some problems such as Poiseuille or Couette flows, edge particles are avoided by using periodic boundary condition. Modelling the fluid flow between two infinite parallel plates can be very expensive and time consuming. Infinte fluid flow and plates can be modelled by assuming vertical boundaries that are periodic in x - direction which allows the fluid particles to interact with the fluid particles at the other side of the boundary and allows particles to flow through the boundary (see Figure 3.17).

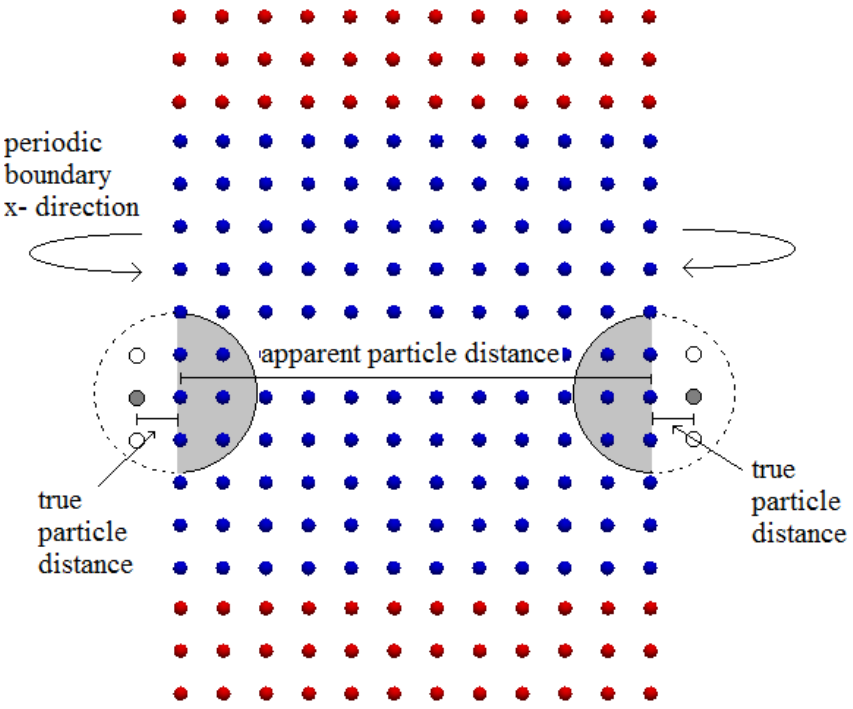


FIGURE 3.17: The periodic boundary condition allows the fluid particles to interact with the fluid particles at the other side of the boundary end even lets to pass through the boundary.

3.11 Time Integration Schemes

There are a few approaches for the time integration used to update the SPH equations in time. Second order schemes are usually used in the SPH literatures such as the Predictor-Corrector scheme[115] and the Verlet scheme[150].

For time integration, it is convenient to rewrite the position (3.101), momentum (3.61), energy (3.71) and density (3.50) equations as:

$$\frac{D\mathbf{r_i}}{Dt} = \mathbf{V_i} \tag{3.77}$$

$$\frac{D\mathbf{v_i}}{Dt} = \mathbf{F_i} \tag{3.78}$$

$$\frac{De_i}{Dt} = E_i \tag{3.79}$$

$$\frac{D\rho_i}{Dt} = D_i \tag{3.80}$$

3.11.1 Predictor-corrector scheme

In the predictor-corrector scheme, the evolution with time is first predicted by taking the following 1/2 step relationships:

$$\mathbf{v}_i^{n+1/2} = \mathbf{v}_i^n + \frac{\Delta t}{2}\mathbf{F}_i^n \tag{3.81}$$

$$\rho_i^{n+1/2} = \rho_i^n + \frac{\Delta t}{2}D_i^n \tag{3.82}$$

$$\mathbf{r}_i^{n+1/2} = \mathbf{r}_i^n + \frac{\Delta t}{2}\mathbf{V}_i^n \tag{3.83}$$

$$e_i^{n+1/2} = e_i^n + \frac{\Delta t}{2}E_i^n \tag{3.84}$$

taking in account that pressure $p_i^{n+1/2}$ depended to the density ($\rho_i^{n+1/2}$) according to Equation (3.72).

The next step consists of correction of the above values at the half step as given below:

$$\mathbf{v}_i^{n+1/2} = \mathbf{v}_i^n + \frac{\Delta t}{2}\mathbf{F}_i^{n+1/2} \tag{3.85}$$

$$\rho_i^{n+1/2} = \rho_i^n + \frac{\Delta t}{2}D_i^{n+1/2} \tag{3.86}$$

$$\mathbf{r}_i^{n+1/2} = \mathbf{r}_i^n + \frac{\Delta t}{2}\mathbf{V}_i^{n+1/2} \tag{3.87}$$

$$e_i^{n+1/2} = e_i^n + \frac{\Delta t}{2}E_i^{n+1/2} \tag{3.88}$$

And then at the end of the time step, the final values for the velocity, density, position and energy are obtained as shown below:

$$\mathbf{v}_i^{n+1} = 2\mathbf{v}_i^{n+1/2} - \mathbf{v}_i^n \tag{3.89}$$

$$\rho_i^{n+1} = 2\rho_i^{n+1/2} - \rho_i^n \tag{3.90}$$

$$\mathbf{r}_i^{n+1} = 2\mathbf{r}_i^{n+1/2} - \mathbf{r}_i^n \tag{3.91}$$

$$e_i^{n+1} = 2e_i^{n+1/2} - e_i^n \tag{3.92}$$

In the end, the pressure p_i^{n+1} is calculated according to Equation (3.72) when the density is (ρ_i^{n+1}).

According to Monaghan [115], this method conserves both angular and linear momentum. In general, instead of using the values at the end of a time step Monaghan used the values at a halfway step, which makes a small error and also saves time.

3.11.2 Verlet scheme

The Verlet algorithm, which is based in the general Verlet approach [150], does not need additional calculations such as predictor and corrector method for each time step, which makes this scheme straightforward. Basically, the values for $\mathbf{v}$, ρ, $\mathbf{r}$ and e

are obtained as written below:

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^{n-1} + 2\Delta t \mathbf{F}_i^n \tag{3.93}$$

$$\rho_i^{n+1} = \rho_i^{n-1} + 2\Delta t D_i^n \tag{3.94}$$

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \Delta t \mathbf{V}_i^n + 0.5\Delta t^2 \mathbf{F}_i^n \tag{3.95}$$

$$e_i^{n+1} = e_i^{n-1} + 2\Delta t E_i^n \tag{3.96}$$

But, at each M time steps (usually, suggested value for M is 50), the variables are calculated using the first order Euler step, given by:

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \Delta t \mathbf{F}_i^n \tag{3.97}$$

$$\rho_i^{n+1} = \rho_i^n + \Delta t D_i^n \tag{3.98}$$

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \Delta t \mathbf{V}_i^n + 0.5\Delta t^2 \mathbf{F}_i^n \tag{3.99}$$

$$e_i^{n+1} = e_i^n + \Delta t E_i^n \tag{3.100}$$

This is necessary to overcome the time integration diverging because the equations are not coupled anymore.

3.12 Particle Positions

In order to get more ordered particle distribution, the so called extended SPH (XSPH) [115] method can be used to update the particles position by replacing Equation (3.77) with the following equation:

$$\frac{d\mathbf{r_i}}{dt} = \mathbf{v_i} + \epsilon \sum_j \frac{m_j}{\rho_{ij}} \mathbf{v_{ji}} W_{ij} \tag{3.101}$$

where $\rho_{ij} = 0.5(\rho_i + \rho_j)$ and ϵ is a constant, ranges between 0 and 1, in most cases $\epsilon = 0.5$.

The XSPH method leads the particle to move with a velocity, which is close to the average velocity of the neighbour particles in the supporting domain by keeping the particles more orderly distributed.

3.12.1 The Time Step

The time step is explicit scheme and chosen according to the Courant-Fredrich-Levy (CFL) condition which, states that the maximum rate of propagation of parameters numerically must not exceed the physical rate [36]. In SPH, it means

$$\Delta t_1 \leq \frac{h}{c_s}. \tag{3.102}$$

where c_s is the reference speed of sound and Δt_1 is the time step to update the SPH equations in time.

If viscosity was taken into account, the minimum time step should take the following form:

$$\Delta t_2 = \min_i \frac{h}{c_s + 0.6(\alpha c_s + \beta \max_{i,j} \mu_{ij})}. \tag{3.103}$$

where α and β are weighting parameters. To be sure that the force exerted on particles are combined correctly, the time step should take the following form:

$$\Delta t_3 = \min_i \left(\sqrt{\frac{h}{|\mathbf{f}_a|}} \right), \tag{3.104}$$

where $\mathbf{f}_a$ is a force per unit mass.

So, a convenient time step is

$$\Delta t = \frac{1}{4} \min(\Delta t_1, \Delta t_2, \Delta t_3). \tag{3.105}$$

The choice of coefficients can be slightly different depending on the model parameters. For example in Monaghan [116] the following was chosen:

$$\Delta t = \min(0.4\Delta t_1, 0.25\Delta t_2, 0.3\Delta t_3). \tag{3.106}$$

For other processes, for instance heat conduction, the time step must be chosen to hold them by using the same arguments.

3.13 The SPH algorithm and Code Structure

The SPH solver was developed from scratch using C++ as a programming language for all problems mentioned in this work. Table 3.1 shows a main overview of the source files used in the SPH solver and the list of the C++ source files can be found in Appendix C. The SPH code is written for one- , two- and three-dimensional problems and can be controlled in common.h file. Also, the constants such as reference density, sound speed, gravity, initial separation and viscosity are defined in common.h source file. After choosing the dimension of the problem, the problem domain is generated with particles carrying properties such as pressure, density, position, velocity, mass, acceleration e.t.c. in the generator.cpp file. In general, the type of the particles is flagged to zero for the fluid particles and one for the solid particles. In the next step, all particles positions are inserted into the kd-tree to find the neighbour particles for each particles in the kdtree.h source file. Further, once the neighbour list completed for all particles, main calculations for the SPH discretised formulations such as updating the pressure with Equation (3.72), momentum with Equation (3.61), velocity and position of the particles with Verlet scheme are carried out in the particle.cpp source file. Further, time is updated and output files are generated with all important parameters or ends program if the simulation time is reached or if not, then again creates new kd-tree and follows the same process as mentioned above.

Table 3.2 shows the main variables that was used in the SPH solver where I and D represent integer and double precisions, respectively. The structure of the code that

was developed is shown in Figure 3.18.

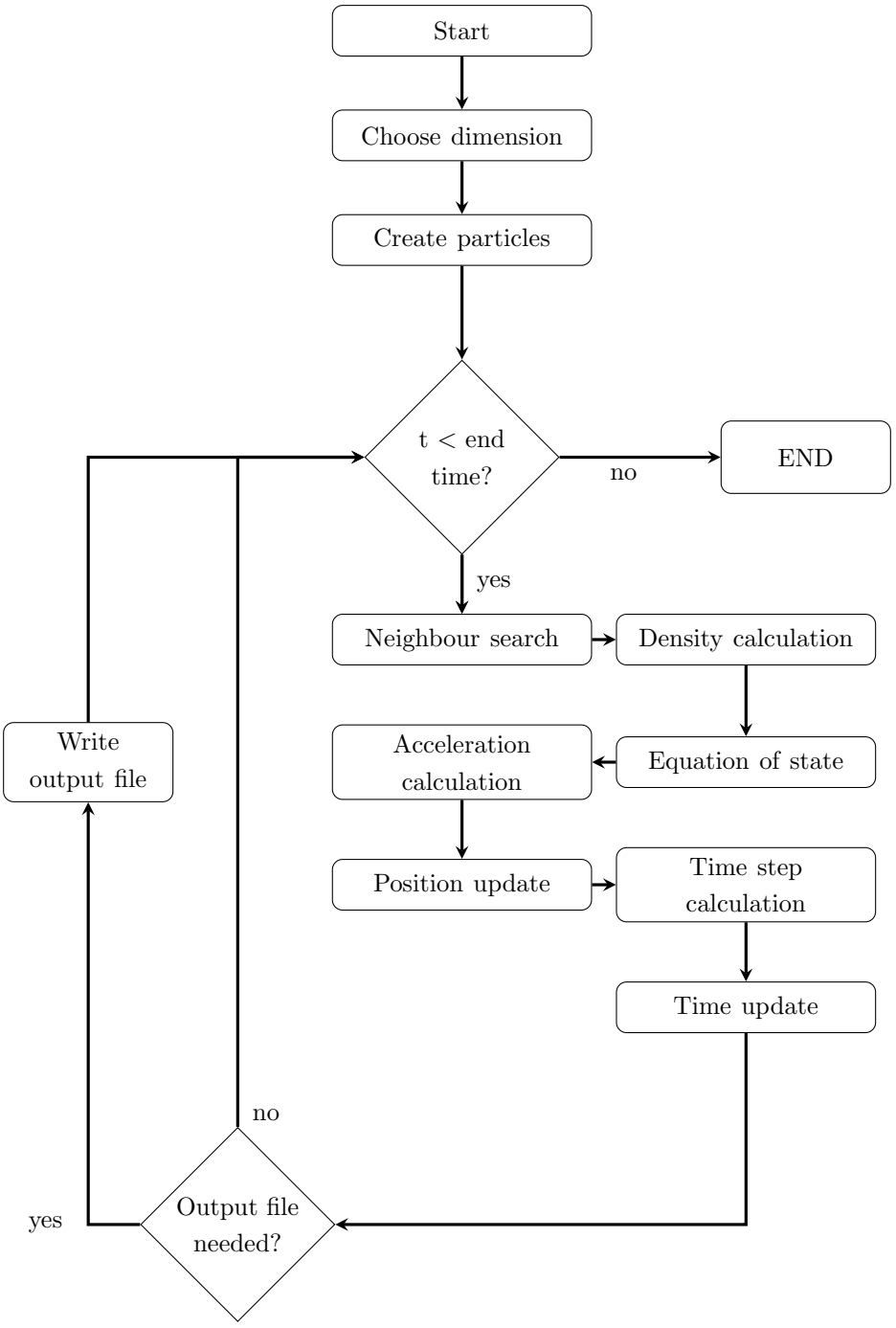
File name	**Descriptions**
main.cpp	Main file used to execute the code
common.h	Defines the constants such as gravity, speed of sound and reference density
generator (.h .cpp)	Generates the problem domain with particles
kdtree.h	Searches for neighbour particles
parameter (.h .cpp)	Defines the domain size, kernel type and time end
particle (.h .cpp)	Calculates all necessary equations such as density and pressure

TABLE 3.1: Source files used in the SPH solver.

Variable name	**Type**	**Descriptions**
m	D	Particle mass
dx	D	Initial separation
t	D	Time
rho	D	Particle density
p	D	Particle pressure
c_0	D	Speed of sound
mu	D	Particle viscosity
h	D	Smoothing length
idx	I	Particle index
v	D	Particle velocity
vel_prime	D	Particle acceleration
$type$	I	Particle type
p_num_x	I	Number of particle in x component
p_num_y	I	Number of particle in y component

TABLE 3.2: Main variables and their descriptions where I and D represent integer and double precisions, respectively.

FIGURE 3.18: The Structure of the code used for simulations



Chapter 4

Validation of SPH methodology

4.1 Perfectly elastic particle bounce

Bouncing of a single particle over a stationary boundary under effect of the gravity was first investigated to determine the accuracy of the kernel implementations and accuracy of the SPH methodology resulted from conservation of mass and momentum; here the momentum interaction is fully invisid and therefore the interaction between the bouncing particle and the boundary is only dependent on the change in pressure gradient. The test case performed is setup similar to the one used by Crespo *et al.* [37].

A schematic of the test case is shown in Figure 4.1. The boundary particles are located in a staggered formation and a single free moving particle is placed a distance, $s = 0.3$ m, away from the boundary. The distances between the stationary boundary particles are dx and $dz/2$ in horizontal and vertical directions respectively, where $dx = dz = h/1.3$ and $h = 2.097 \cdot 10^{-2}$ m. The single free moving particle falls under influence of gravity from rest at an initial height $s = 0.3$ m. The sound speed was chosen to be 50 times maximum velocity during the simulation which is equal to the velocity of the bouncing particle just before hitting the boundary particles to limit the compressibility of the material. All SPH particles are modelled using the cubic

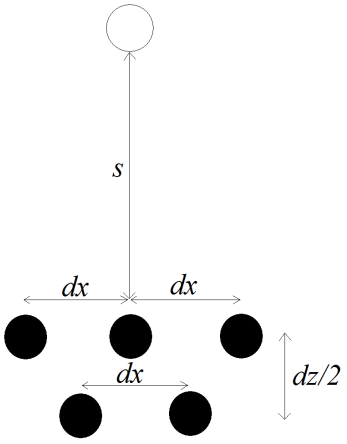


FIGURE 4.1: Schematic of stationary boundary particles positions. The unshaded circle describes the bouncing single particle and shaded circles describe boundary particles

spline kernel (Equation 3.33) and the Equation of state (Equation 3.72).

Only gravitational force acts on the moving particle until it is close to smoothing length, $2h$, to the boundary where the falling particle starts interacting with the boundary particles. Figure 4.2 shows the changes in density and pressure when the distance between the free moving particle and boundary diminishes. The density of the falling particle increases according to Equation (3.50), which leads an increase in pressure from Equation (3.72). This creates repulsion force. Direction of the force due to the pressure term is in opposite direction to the falling particle and this can be explained with negative sign in discretised momentum equation (see Equation (3.54)) due to the pressure. This repulsion force is enough to stop the falling particle and repel in opposite direction with the same velocity.

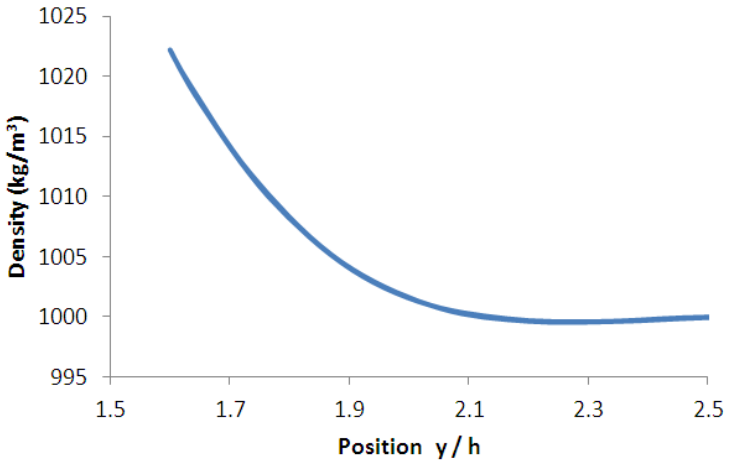
Figure 4.3 shows the change of height of the free moving particle against time; there is no considerable difference between the maximum height after bouncing and the initial height with a maximum error of 0.2% found at the return position. Figure 4.4 shows the change in velocity at y-axis against time. Here, velocity of the bouncing particle before collision is equal to the velocity after collision but in the opposite direction, which is the result of the force exerted from the stationary boundary particles and thus conserves momentum. The Figure 4.5 shows the velocity change against height using the SPH method (blue crosses) and in good agreement with analytical solutions

(red line) obtained from kinematics equations derived from the following relationships:

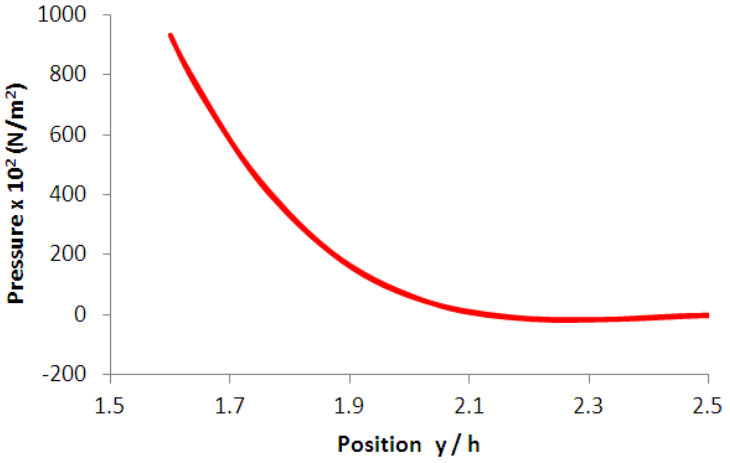
$$v = \sqrt{2gh} \tag{4.1}$$

where v is the velocity, g is the gravity and h is the height of the free moving particle.

This simple test shows that bouncing particle can be returned to the initial height without losing energy due to the repulsive force between particles.



(A)



(B)

FIGURE 4.2: Change of density and pressure of the bouncing particle close to boundary particles

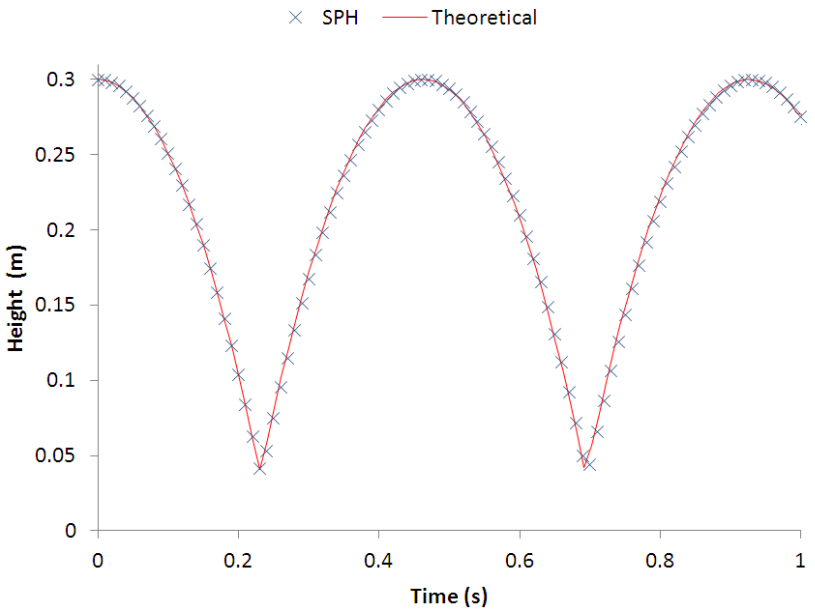


FIGURE 4.3: Change of height of the bouncing single particle against time and the SPH results are in good agreement with the theoretical results

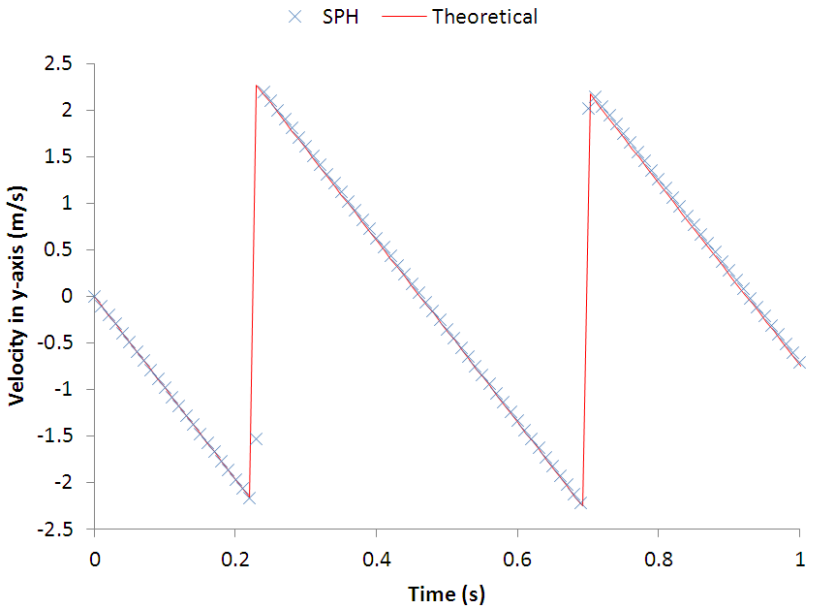


FIGURE 4.4: Change of velocity in y-axis of the bouncing single particle against time and the SPH results are in good agreement with the theoretical results

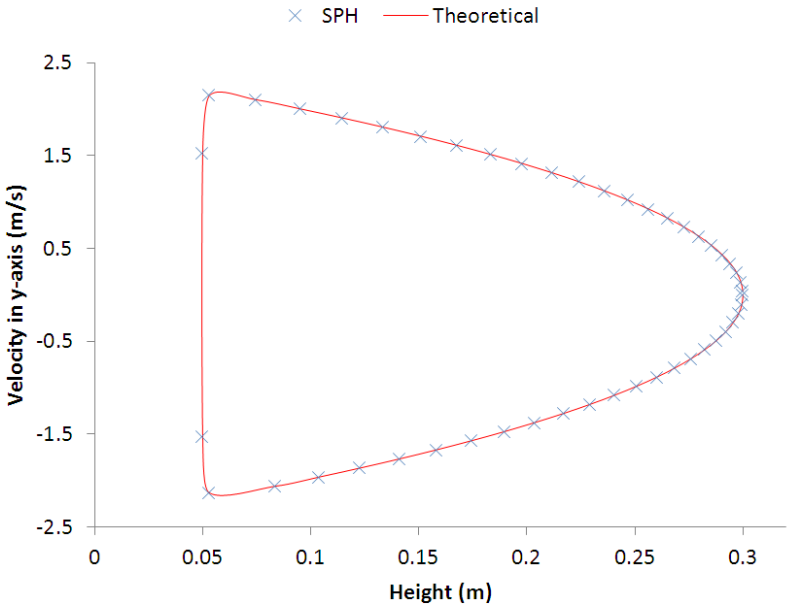


FIGURE 4.5: Change of velocity in y-axis of the bouncing single particle against height and the SPH results are in good agreement with the theoretical results

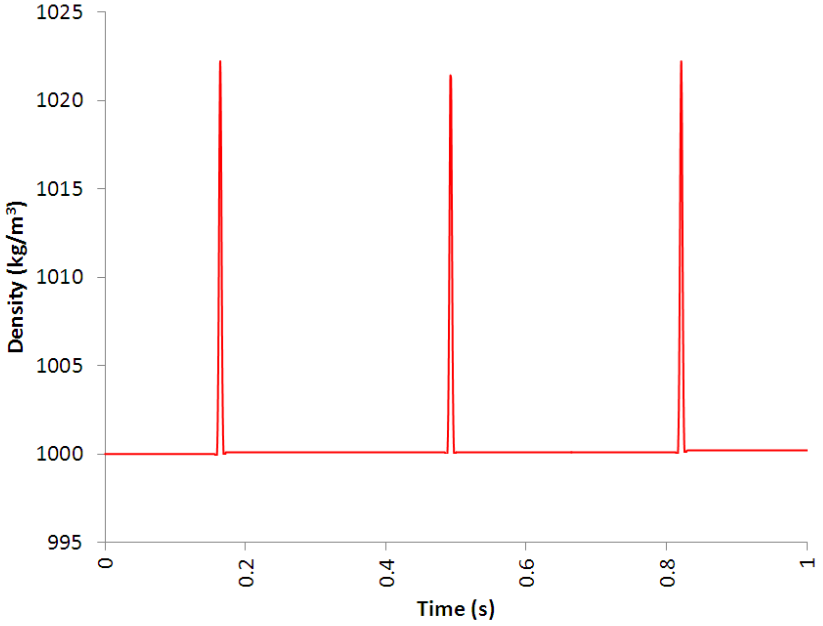


FIGURE 4.6: Change of density of the bouncing single particle against time and the SPH results are in good agreement with the theoretical results

4.2 Couette flow

A two-dimensional Couette flow is used to test the effects of viscous dissipation where a fluid is driven between two infinite parallel plates, one of which is moving while the other is stationary. The plates are separated by a distance of $L = 10^{-3}$ m. Periodic boundary conditions are applied in the x direction so that SPH particles to mimic an infinitely long domain. The initial conditions of the fluid are same with those used by Morris[123], as shown in Table 4.1, and the fluid used is water.

The lower plate is kept static whilst the upper plate is moving with a constant velocity of $V_0 = 1.25 \cdot 10^{-5}$ m/s and this drives the fluid which was initially at rest and a schematic of the Couette flow is shown in Figure 4.7. All SPH particles are separated from each other by $dx = dy$ in x and y axis. A cubic kernel is used with smoothing length of $h = 1.3dx$. The plates are modelled using three layers of particles with constant density and is equal to the initial density of the fluid particles. The speed of sound is chosen as $c_0 = 100U_{max}$. With U_{max} equal to the velocity of the moving plate V_0. The velocity difference between the fluid and the moving plate created viscous shear forces between the layers of fluid particles.

	Fluid	**Lower plate**	**Upper plate**
Pressure, $p\ (Pa)$	0	0	0
Velocity, $\vec{v}\ (m/s)$	0	0	0
Acceleration, $\dot{\vec{v}}\ (m/s^2)$	0	0	0
Initial particle separation, $dx = dy\ (m)$	4.54×10^{-5}	4.54×10^{-5}	4.54×10^{-5}
Smoothing length, $h\ (m)$	$1.3dx$	$1.3dx$	$1.3dx$

TABLE 4.1: The initial conditions of the SPH particles for Couette flow

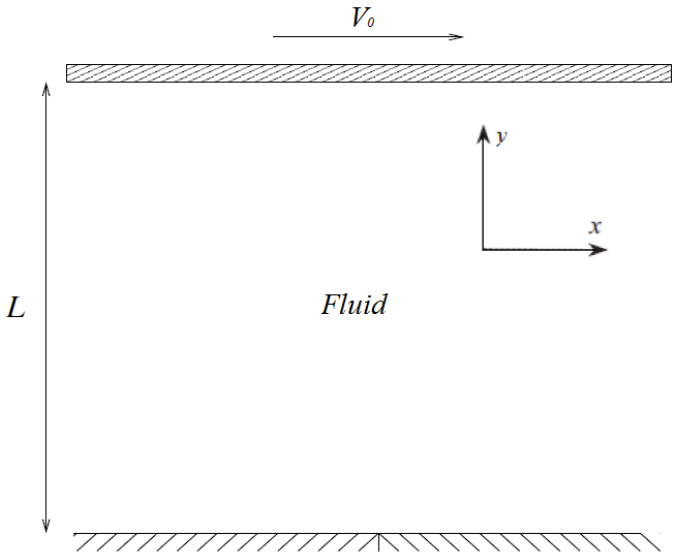


FIGURE 4.7: The geometry of the Couette flow in 2D.

The theoretical solution of the Navier-Stokes equations used for comparison is obtained from Morris *et al.* [123] is:

$$v_x(y,t) = \frac{V_0}{L}y + \sum_{n=1}^{\infty} \frac{2V_0}{n\pi}(-1)^n \sin\left(\frac{n\pi}{L}y\right) \exp\left(-\nu\frac{n^2\pi^2}{L^2}t\right). \tag{4.2}$$

where v_x is the velocity of the fluid in x - axis, ν is kinematic viscosity, ρ is the density of the fluid and L is the distance between lower and upper plates.

Figure 4.8 shows the comparison of the velocity profile obtained from the Equation (4.2) and SPH at different times. The steady-state results are in good agreement with maximum error of 0.5%, verifying the approach used for viscous forces with SPH method. The same test case was simulated with SPH by Morris [123] and the results were in close agreement with error of within 0.5%.

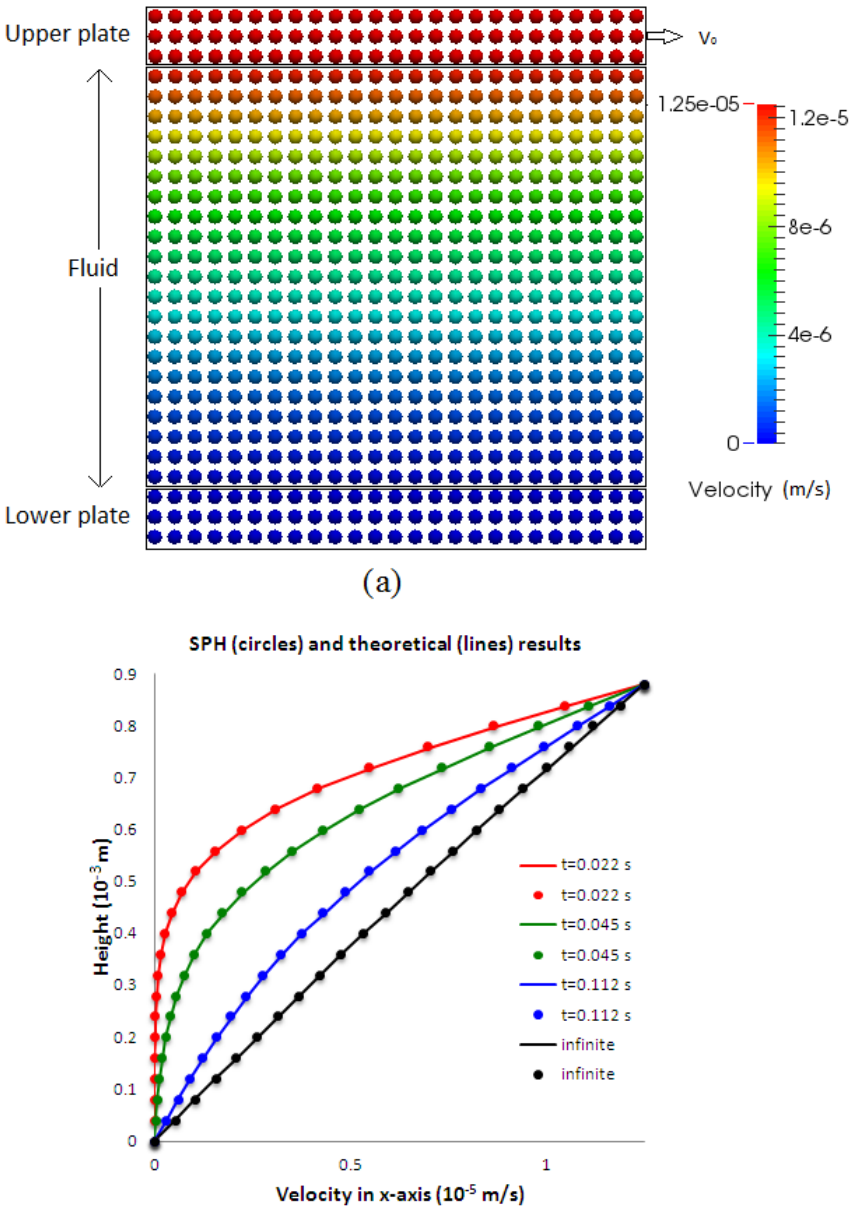


FIGURE 4.8: a) Velocity field for Couette flow simulation with SPH particles distribution at the time when the fluid gets to steady state motion. b) Comparison of theoretical and the SPH solutions at four different times, and the results are in good agreement with maximum error of 0.5%

4.3 Poiseuille flow

A two-dimensional Poiseuille flow problem is also used to examine the viscous effects applied by a pressure gradient force on a fluid located between two stationary infinite plates. Following Morris [123], a body force in the x direction is applied to simulate pressure gradient and periodic boundary conditions are applied to create an infinitely long domain in the x direction.

The initial conditions of the fluid are same with those used by Morris [123], shown in Table 4.2, and the fluid used is water. Lower and upper plates are separated from each other by a distance of $L = 10^{-5}$ m, as shown in the schematic in Figure 4.9. All SPH particles are separated from each other by $dx = dy$ in x and y axis. A cubic kernel is used with smoothing length of $h = 1.3dx$. The plates are modelled using three layers of particles with constant density and is equal to the initial density of the fluid particles. The speed of sound is chosen as $c_0 = 10^{-3}$ m/s.

Figure 4.10a shows the steady-state SPH results, demonstrating good agreement for maximum velocity of $1.23 \cdot 10^{-5}$ m/s compared against $1.223 \cdot 10^{-5}$ m/s calculated analytically. A theoretical solution for comparing the velocity profile at various times

	Fluid	**Lower plate**	**Upper plate**
Pressure, $p\ (Pa)$	0	0	0
Velocity, $\vec{v}\ (m/s)$	0	0	0
Acceleration, $\dot{\vec{v}}\ (m/s^2)$	10^{-4}	0	0
Initial particle separation, $dx = dy\ (m)$	1.96×10^{-7}	1.96×10^{-7}	1.96×10^{-7}
Smoothing length, $h\ (m)$	$1.3dx$	$1.3dx$	$1.3dx$

TABLE 4.2: The initial conditions of the SPH particles

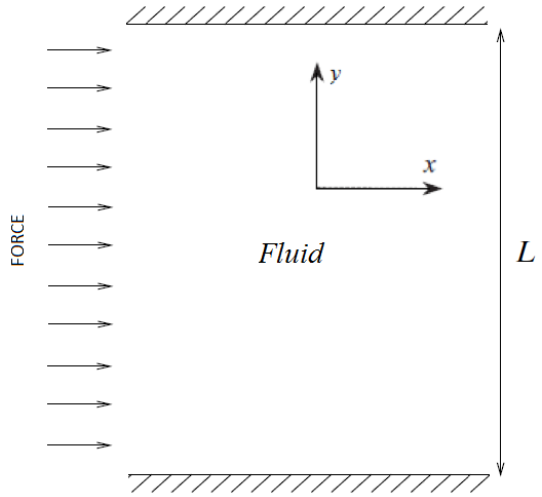


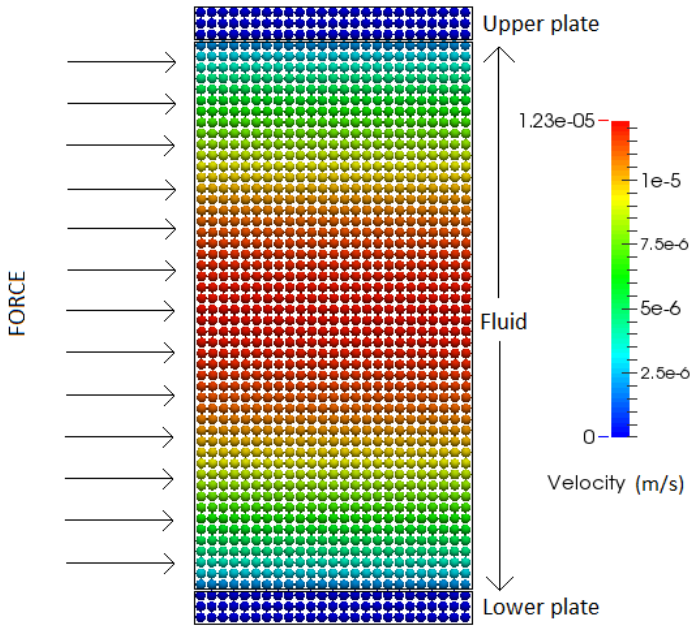
FIGURE 4.9: The geometry of the Poiseuille flow in 2D.

is given by:

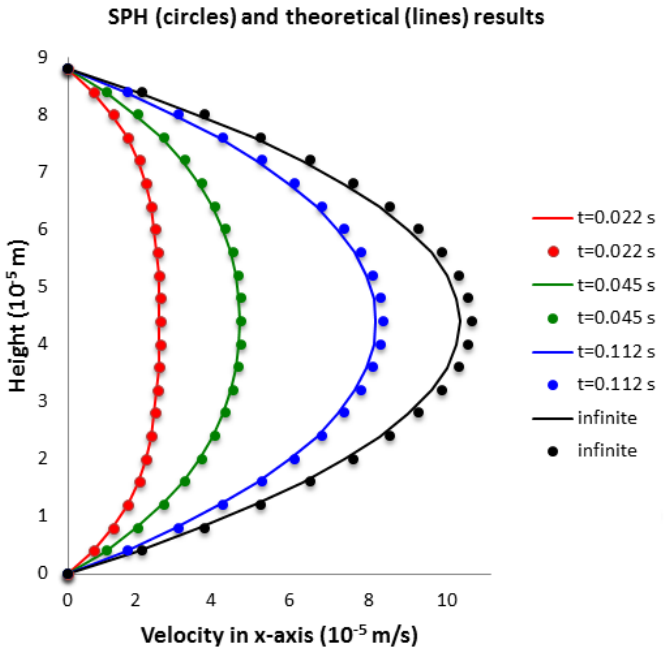
$$v_x(y,t) = \frac{F}{2\nu} y(L-y) - \sum_{n=0}^{\infty} \frac{4FL^2}{\nu\pi^3(2n+1)^3} \sin\left(\frac{\pi y}{L}(2n+1)\right) \exp\left(-\frac{(2n+1)^2\pi^2\nu}{L^2}t\right). \tag{4.3}$$

where v_x is the velocity of the fluid in x - axis, ν is viscosity, ρ is the density of the fluid, L is the distance between lower and upper plates.

The SPH solution exhibits a maximum error in profile at steady-state of 0.5% and is comparable to those presented in Morris [123] and Violeau [151]. Figure 4.10 shows the comparison of the velocity profile obtained from the Equation (4.3) and SPH at different times. High accuracy is seen with maximum error of 0.5% between SPH and the theoretical solution. The same test case was simulated with SPH by Morris [123] and the results were in close agreement with error of within 0.7%.



(a)



(b)

FIGURE 4.10: a) Velocity field for Poiseuille flow simulation with SPH particles distribution at the time when the fluid gets to steady state motion. b) Comparison of theoretical and the SPH solutions at four different times.

4.4 Lid Driven Cavity

The capability of SPH to model high Reynold number flows is considered by using well-defined lid-driven cavity test case. Here the fluid is located in a rectangular enclosed cavity with dimensions $0 \leq x \leq 1$ and $0 \leq y \leq 1$ with a moving lid. The geometry of the square cavity is as shown in Figure 4.11. The lid moves with a constant velocity V_0 to the right. Due to the effects of viscous forces, the fluid will begin rotating clockwise in the square cavity, and the developed flow structure is dependent only on the velocity of the lid. Water is used as the fluid. The particles are initially located at a distance $dx = dy$ from each other in both the vertical and horizontal directions, and their velocity and pressure set at zero. A cubic kernel is used with smoothing length of $h = 1.3dx$. The walls of the square cavity are modelled using three layers of stationary particles with constant density and is equal to the initial density of the fluid particles. The speed of sound is chosen as $c_0 = 100U_{max}$. With U_{max} equal to the velocity of the lid V_0.

The lid-driven cavity problem is solved for three different Reynolds numbers, $Re = 100$, $Re = 1000$ and $Re = 10,000$, each with resolutions of 50×50, 100×100

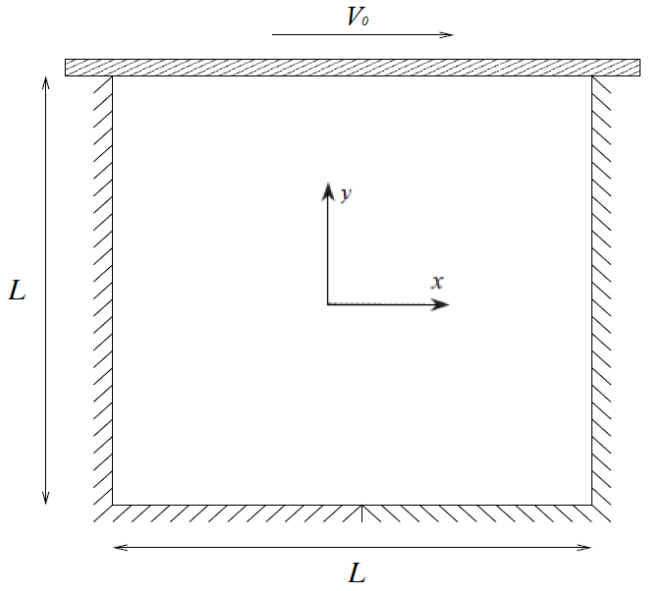
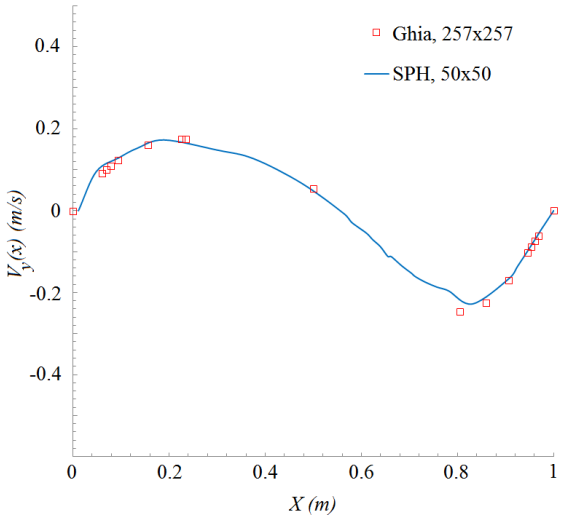


FIGURE 4.11: The geometry of the lid driven cavity flow in 2D

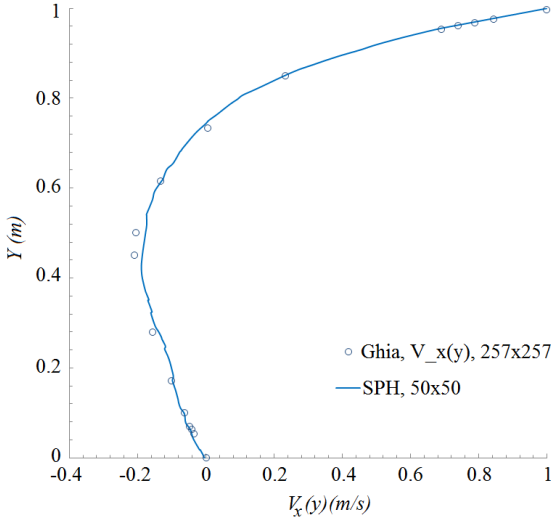
and 200×200 particles, respectively. There is no analytical solution for the lid-driven cavity, but there exists numerical solution for comparison and validation. The results are compared to the finite volume approach performed by Ghia *et al.* [61], who modelled the lid-driven cavity case with multi-grid on a 257×257 mesh and Adami *et al.* [2] who modelled the problem using a transport-velocity formulation with weakly-comressible SPH. Comparisons are made using the steady-state flow results.

Figure 4.12 shows the steady-state velocity profile for $Re = 100$. The velocity field (see Figure 4.12c) is in good agreement with the velocity field (Figure 4.12d) which simulated by Adami *et al.* [2]. The colors in the Figures 4.12c and 4.12d represents the absolute value of the velocity changing from zero (blue) to V_{LID} (red), and the fluid rotates in the clockwise direction. The fluid close to the bottom of the square cavity moves very slow and due to the shear force at the moving lid, a single core vortex appears between the centre of the cavity and the moving lid. Figure 4.12a and 4.12b shows the comparison of the results with Ghia *et al.* [61], where the horizontal and vertical velocity components are plotted against the vertical and horizontal centreline of the cavity, respectively. It shows that for the SPH approach, resolution of 50×50 particles provides sufficient accuracy for low Reynolds number, $Re = 100$, lid driven cavity flows.

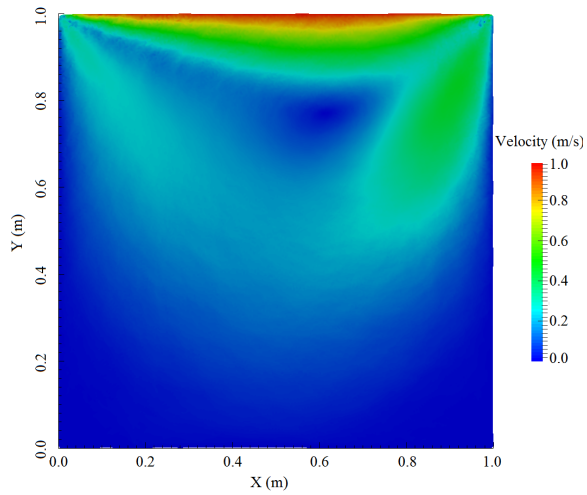
As the velocity of the lid increases for $Re = 1000$ and $Re = 10000$, see Figures 4.13c and 4.14c, the velocity field showed that the size of the core vortex becomes larger with increasing Reynolds number and is located more centrally. At $Re = 1000$, accuracy increases with increasing the particle resolution as shown in Figures 4.13a and 4.13b and the velocity field (see Figure 4.13c) is in good agreement with the velocity field (see Figure 4.13d) which simulated by Adami *et al.* [2]. The lid driven cavity test shows that our developed solver can be used to simulate fluid flows with high Reynolds number.



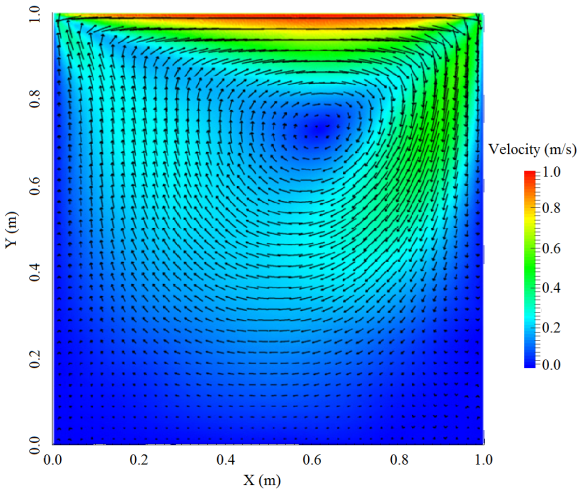
(A) Velocity profile: $V_y(x)$ at $y = 0.5$ (vertical centreline)



(B) Velocity profile: $V_x(y)$ at $x = 0.5$ (horizontal centreline)

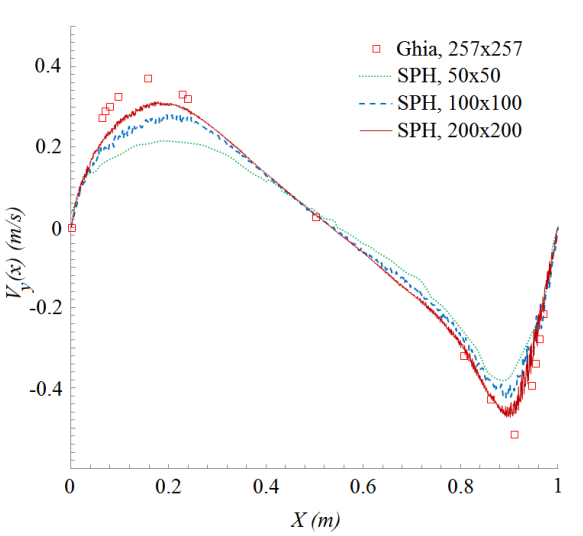


(C) Velocity field with vectors. 50x50 particles

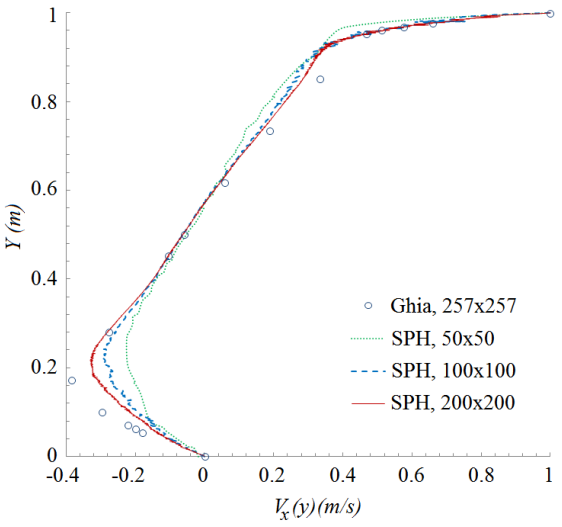


(D) Similar velocity field obtained by Adami *et al.* [2]

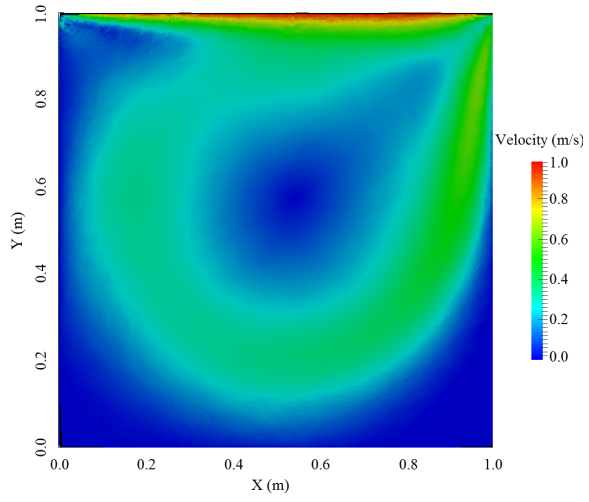
FIGURE 4.12: Lid Driven Cavity, Re=100



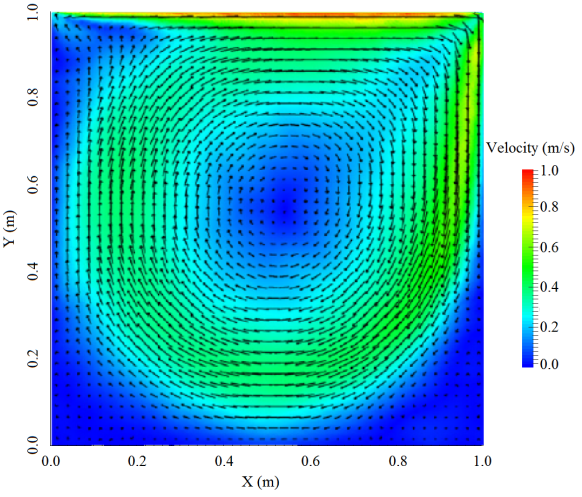
(A) Velocity profile: $V_y(x)$ at $y = 0.5$ (vertical centreline)



(B) Velocity profile: $V_x(y)$ at $x = 0.5$ (horizontal centreline)

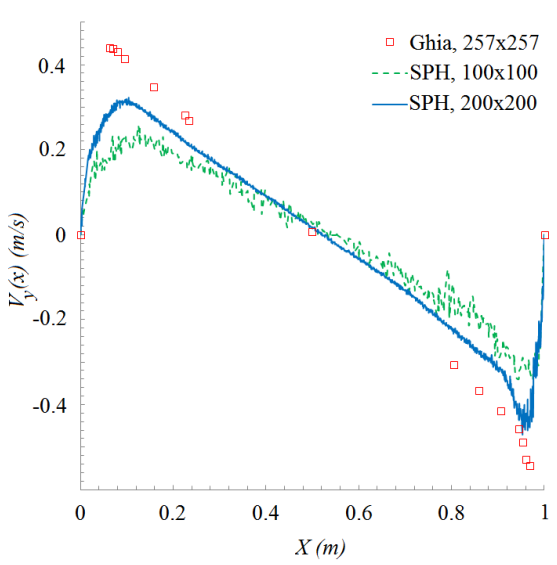


(C) Velocity field with vectors. 200x200 particles

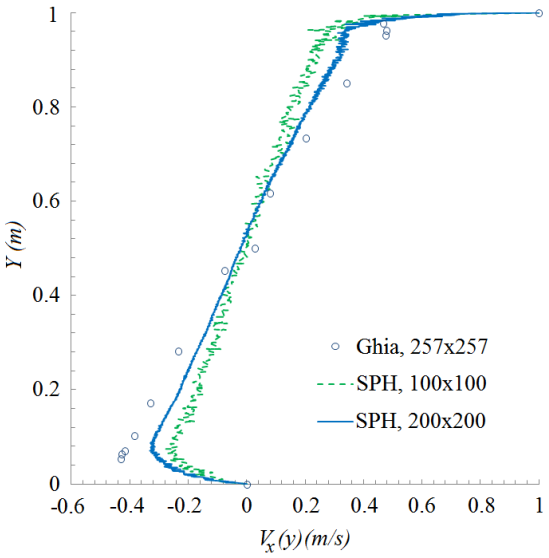


(D) Similar velocity field obtained by Adami *et al.* [2]

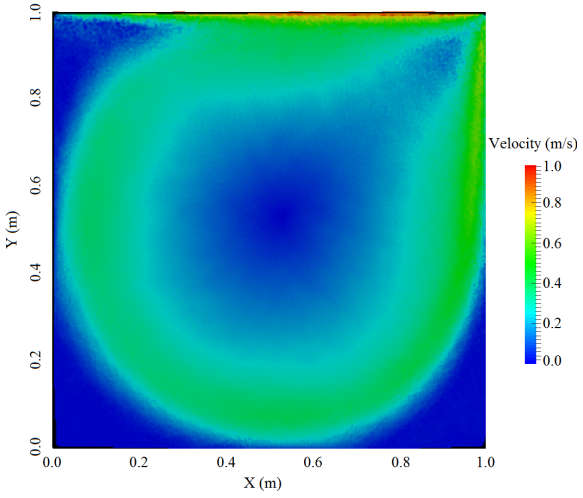
FIGURE 4.13: Lid Driven Cavity, Re=1000



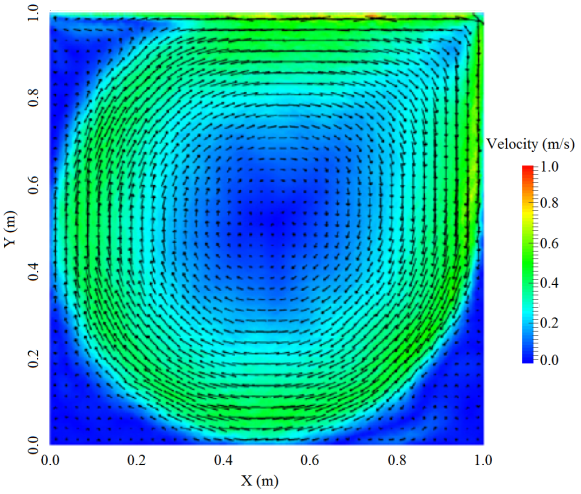
(A) Velocity profile: $V_y(x)$ at $y = 0.5$ (vertical centreline)



(B) Velocity profile: $V_x(y)$ at $x = 0.5$ (horizontal centreline)



(C) Velocity field with vectors. 200x200 particles



(D) Similar velocity field obtained by Adami *et al.* [2]

FIGURE 4.14: Lid Driven Cavity, Re=10000

4.5 Dam break over a dry tank

Previous test cases such as the Couette flow, Poiseuille flow and lid driven cavity flows are well defined as the domain is fully enclosed and as a result, fluid particles are surrounded by boundary particles which avoids density deficiency near the boundaries that ensured the availability of a full kernel support. 2D Dam break test case is considered to study the behaviour of the particles with a free surface. The findings are compared against the collapse of a dam over a dry tank due to the effects of gravity against the experiment done by Koshizuka and Oka [80]. A full explanation of the experiment is available in Koshizuka and Oka [80] with a schematic configuration shown in Figure 4.15. The tank is $4L = 0.584$ m long, with initial size of the water column $L = 0.146$ m long and its height $2L = 0.292$ m. The fluid used is water.

Water particles are initially set at rest, with initial particles separation that is uniformly equi-distance, $dx = dy = L/25$. A cubic kernel is used with smoothing length

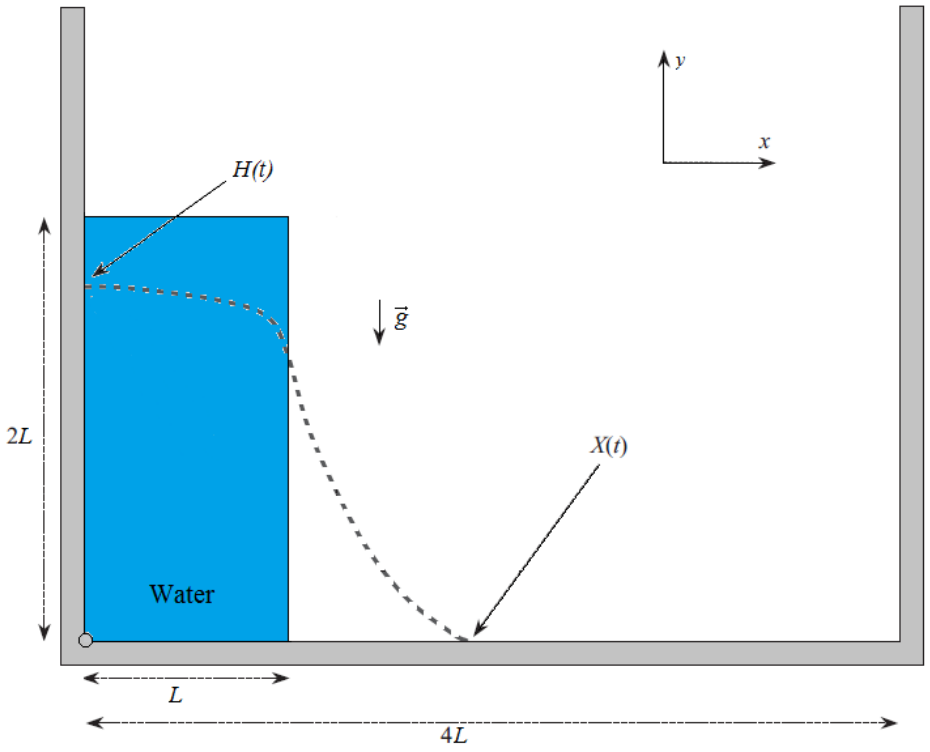


FIGURE 4.15: The schematic configuration of the water is shown by a blue color and subsequent configuration by a dashed line.

of $h = 1.3dx$. The walls of the tank are modelled using three layers of dummy particles using the same parameters as the fluid particles and a repulsive force (see Equation (3.76)) is applied to prevent particle penetration.

In this simulation, the stabilized continuity density Equation (3.51) is used rather than traditional continuity density Equation (3.50) due to its advantages of removing spurious oscillations in the density field. The comparison of the density fields between the continuity Equation (3.50) and the stabilized continuity Equation (3.51) is shown in Figure 4.16 at different times. It can be seen that the stabilized continuity equation reduces the density fluctuation significantly which in its turn provides much smoother pressure and velocity fields as shown in Figure 4.17 and 4.18, respectively.

It is obvious that the stabilized continuity density approach produces smoother and more realistic free surface compared to the traditional continuity density approach as shown in Figure 4.19. Also, it must be underlined that the current approach avoids the very disordered particles as it can be seen in the right column in Figure 4.16, but it does not ideally distribute particles ordered (sometimes two particles may come too close to each other i.e. see Figure 4.16) and it may require extra treatment to make particles movement more homogeneous, such as XSPH [117].

Dehnen and Aly [43] concluded in their joint work that Wendland [155] kernel functions (see Equation (3.39)) are ideal candidate for free surface problems. Figure 4.20 shows the comparison between the Wendland kernel function and the cubic kernel function. In the left column in Figure 4.20, red circles highlight some regions two neighbour particles may come too close to each other with the cubic kernel function despite of using the stabilized continuity density approach. However, the Wendland kernel function keeps the distance between neighbour particles constant as it shown in right column of Figure 4.20. The minimum inflection point for the gradient in the Wendland kernel occurs at $q = 0.5$ compared to other kernels, for example, the cubic spline kernel where the minimum inflection point is $q = 0.75$. From the SPH discretised Navier-Stokes equation (see Equation 3.61) the pressure gradient term is influenced by ∇W and the wider positive gradient range is delineated by the dash-red

line of the Wendland kernel (between 0.5 and 2.0) as shown in Figure 3.9; while for other kernels, for example, the cubic spline kernel has a range between 0.75 and 2.0. This allows inter-particles repulsion to occur over a wider range (when $q \geq 0.5$) before diminishing returns after the inflection point (when $q < 0.5$).

As shown in Figure 4.21, the x-position of the leading edge, Z, which is also known as a dam toe in SPH literatures and the maximum dam depth, H, on the left-hand wall were considered over time, t, in order to validate the model. The comparison between the SPH results and the experimental results provided by Koshizuka and Oka [80] showed very close agreement, where solid lines correspond to the SPH results and dots to experimental results. The small offset along, Z, which means that the speed of the dam toe is slower than the SPH calculations and this could be due to the friction force between the tank and the fluid.

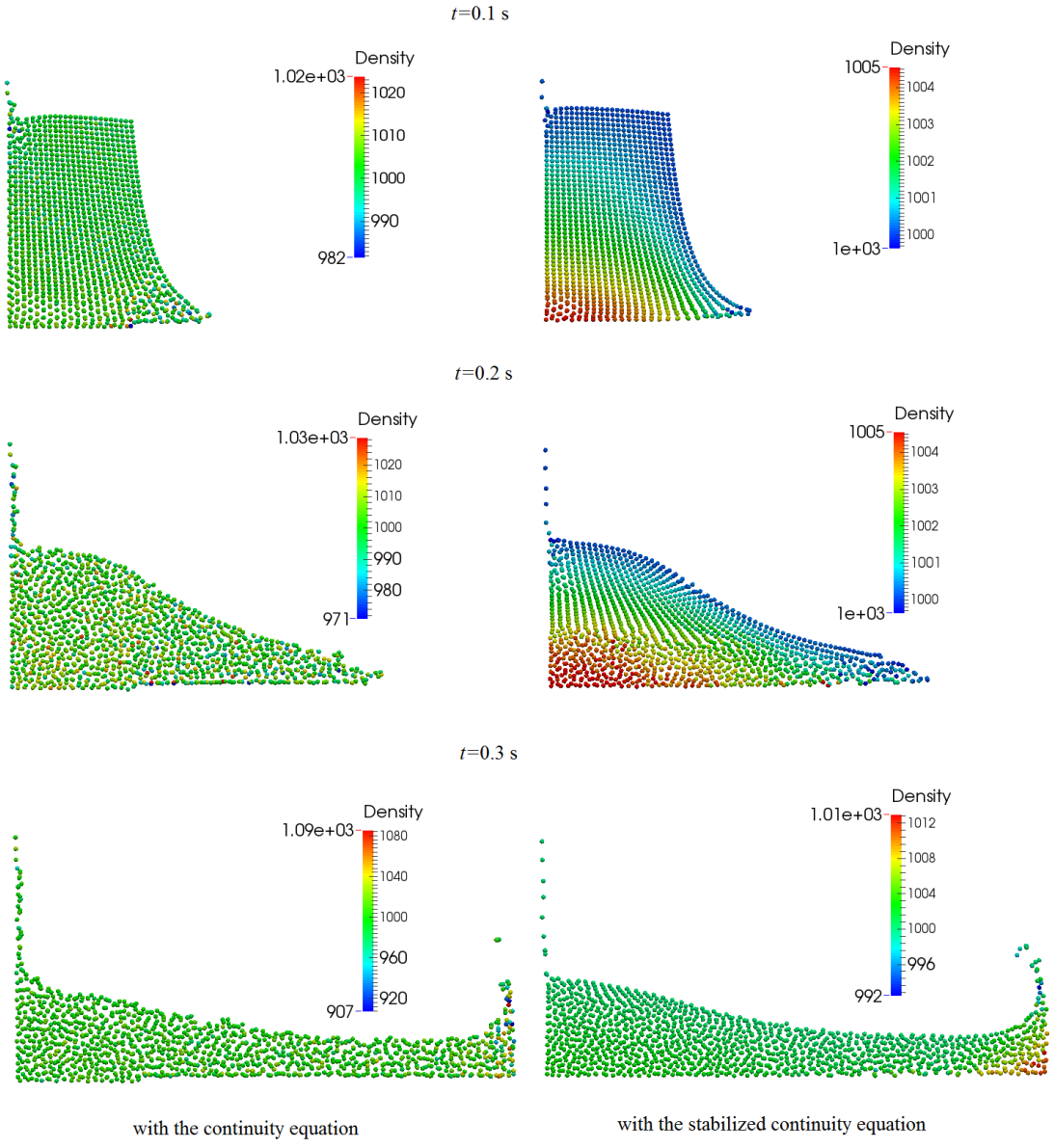


FIGURE 4.16: The SPH density fields comparison between the traditional continuity density approach (left column) and the stabilized continuity density (right column) at $t = 0.1$ s, $t = 0.2$ s and $t = 0.3$ s. Densities are in kg/m^3

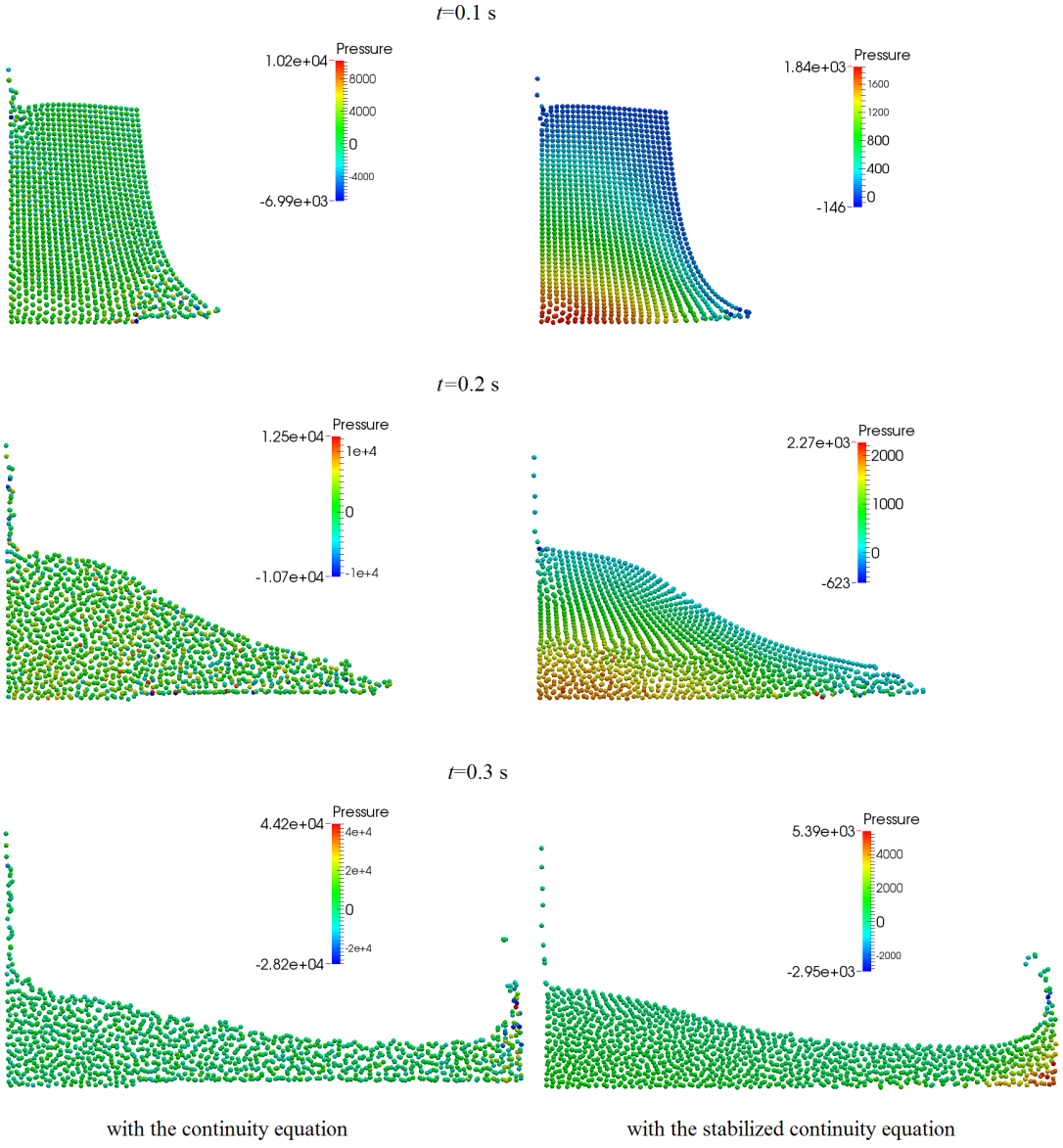


FIGURE 4.17: The SPH pressure fields comparison between the traditional continuity density approach (left column) and the stabilized continuity density (right column) at $t = 0.1$ s, $t = 0.2$ s and $t = 0.3$ s.

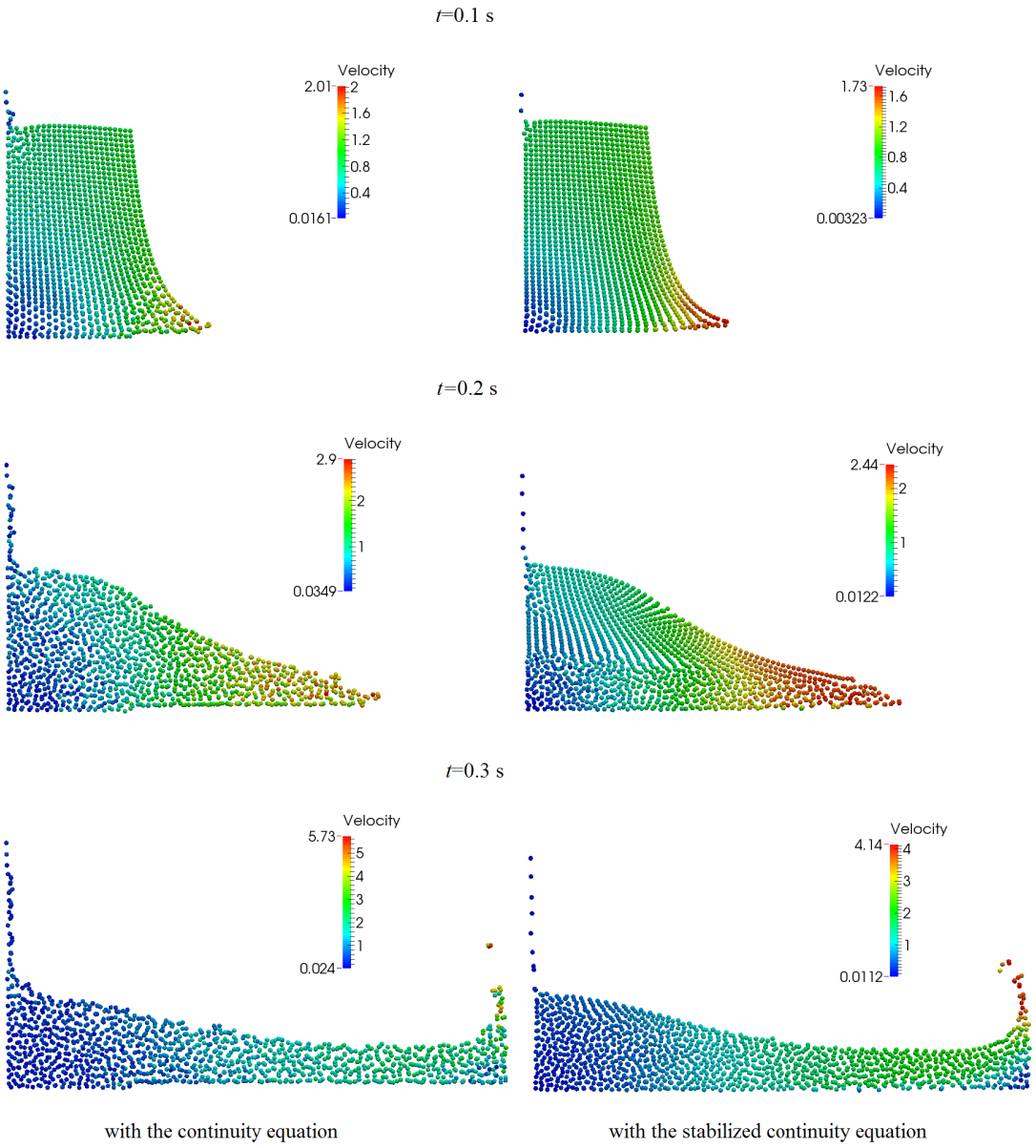


FIGURE 4.18: The SPH velocity magnitude fields comparison between the traditional continuity density approach (left column) and the stabilized continuity density (right column) at $t = 0.1$ s, $t = 0.2$ s and $t = 0.3$ s. Velocities are in m/s

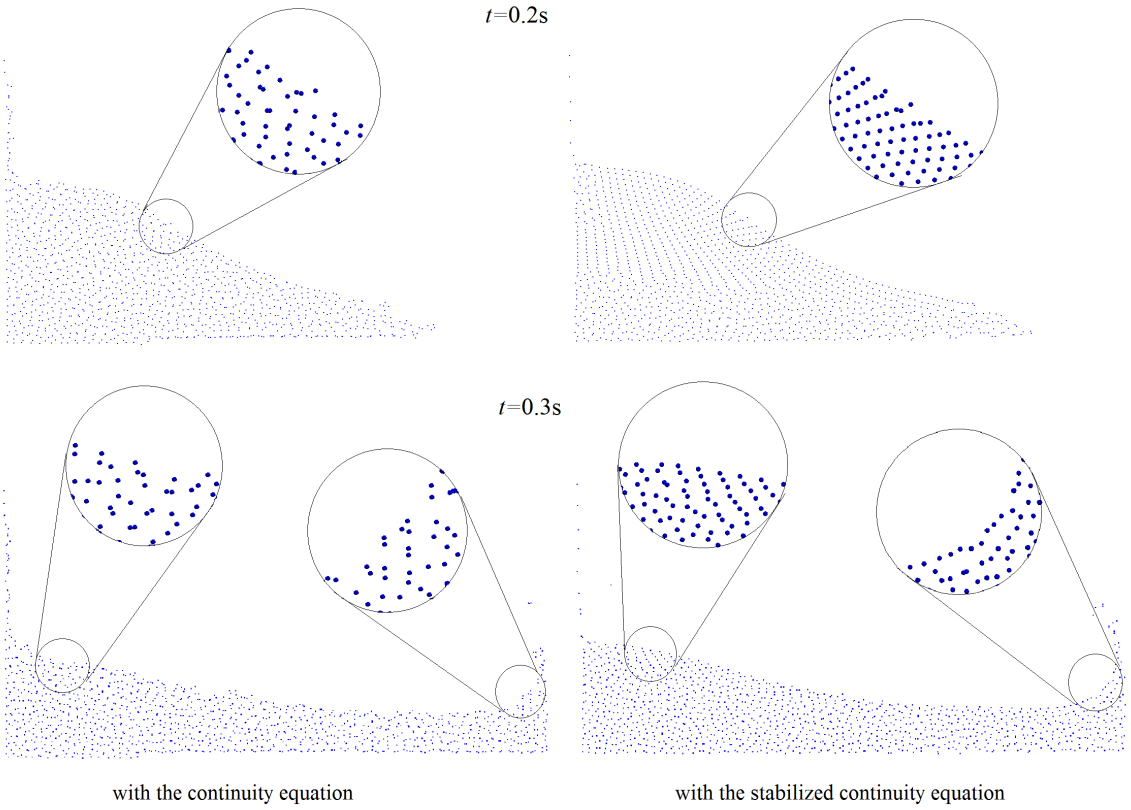


FIGURE 4.19: The SPH free surface comparison between stabilized continuity density approach and traditional continuity density approach at $t = 0.2$ s and $t = 0.3$s

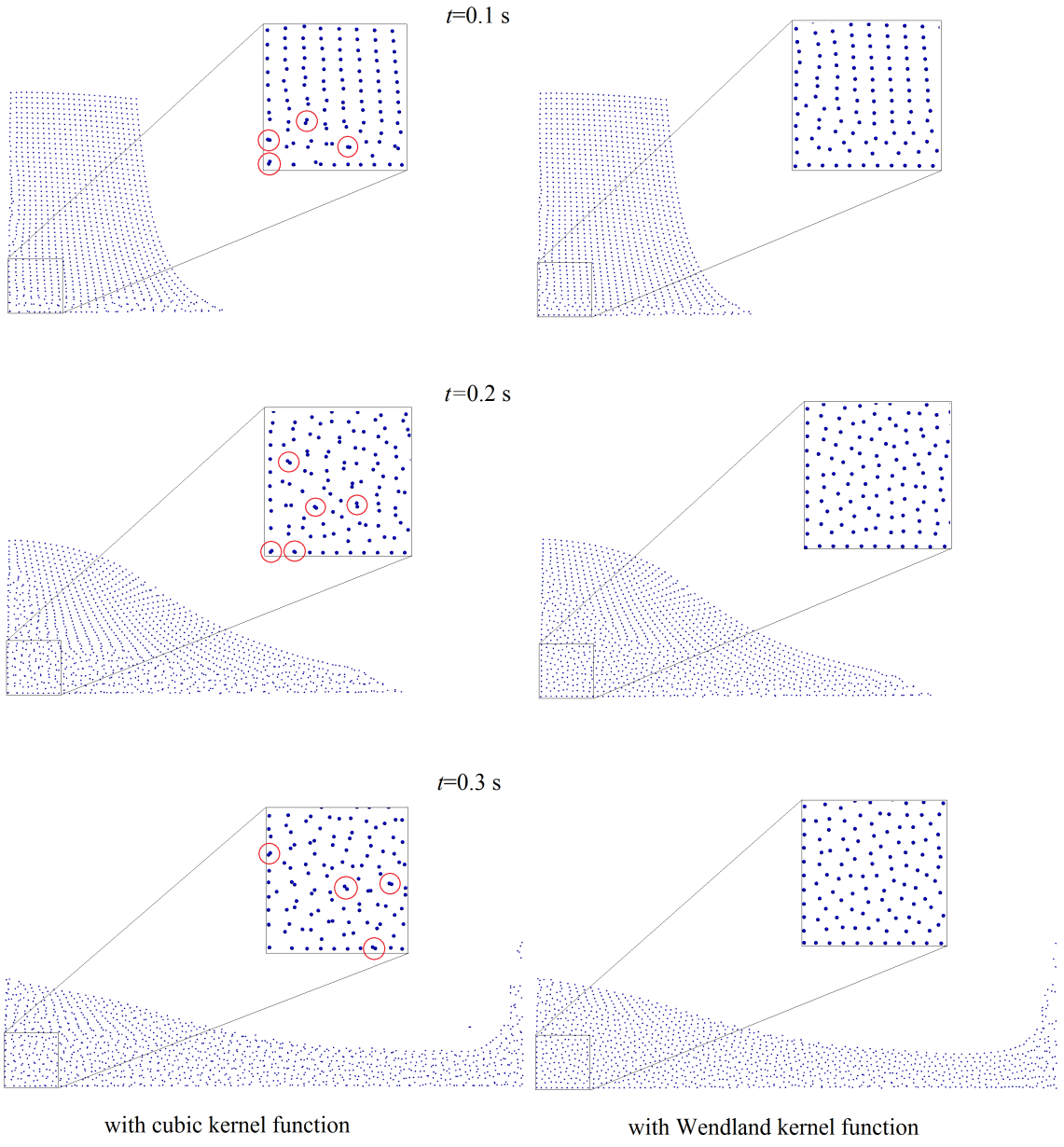


FIGURE 4.20: The SPH comparison between the cubic kernel function (left column) and the Wendland kernel function (right column) at $t = 0.1$ s, $t = 0.2$ s and $t = 0.3$s. Some particle clusterings are shown with red circles

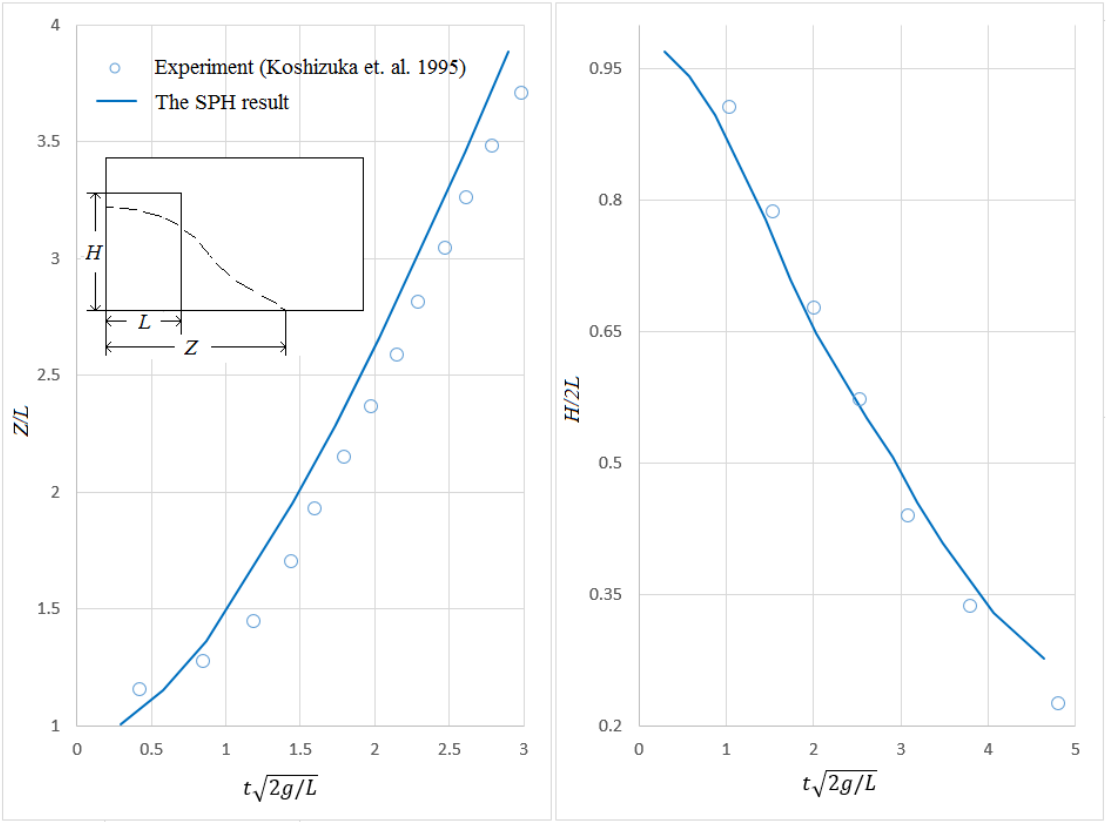


FIGURE 4.21: Dam break over a dry tank. Non-dimensional maximum x-position of the dam toe Z/L and the maximum dam depth $H/2L$ on the left-hand wall versus non-dimensional time $t\sqrt{2g/L}$. Solid lines correspond to the SPH results and dots to experimental results [80].

Chapter 5

Surface Tension

5.1 Surface Tension

The surface tension force is considered as an external force acting on the free surface of the fluid. Ordinarily, the Navier-Stokes equation does not contain the surface tension force and it is considered as an external force. The boundaries of Lagrangian fluid can be defined by the particles. Fluid molecules inside the fluid are in absolute equilibrium due to the attractive forces among the all neighboring molecules. The attractive forces are not balanced for the fluid molecules at the surface of the fluid and this produces surface tension.

In general, there are two methods to model the effect of surface tension with SPH. In the first method, an inter - particle interaction force (IIF) is introduced between all SPH particles to simulate the effect of the surface tension directly [75, 101, 102]. Tartakovsky *et al.* [145] suggested the following equation for the inter-particle interaction force:

$$\mathbf{F}_{ij} = \begin{cases} s_{ij} \cos\left(\dfrac{1.5\pi}{3h}|\mathbf{r}_{ij}|\right) \dfrac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|}, & |\mathbf{r}_{ij}| \leq 3h \\ 0, & |\mathbf{r}_{ij}| > 3h \end{cases} \tag{5.1}$$

where s_{ij} is the strength of the force between particles and also controls the type of the fluid.

It can be seen from the above equation that it is repulsive when particles comes close to each other and attractive when particles move further apart as shown in Figure 5.1. These forces are expected to be balanced inside the droplet due to the "symmetrical" distribution of the particles. On the other hand, it is not balanced at the free surface and the resultant direction of the net force is in the same direction of the surface normal. These net forces on the surface contribute to the surface tension forces.

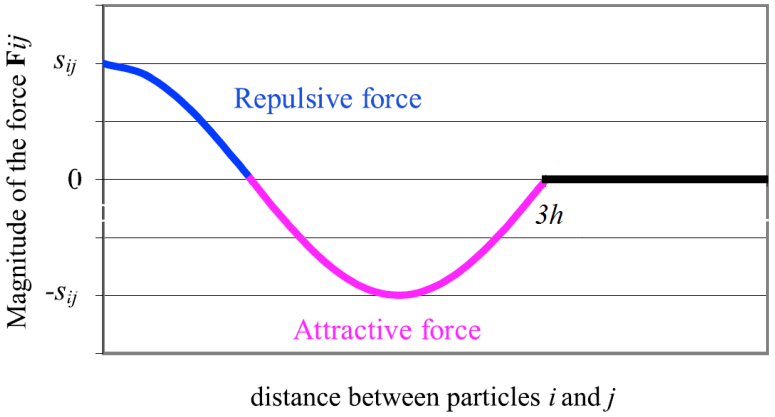


FIGURE 5.1: Inter-particle interaction force is repulsive when particles are close to each other and attractive when particles are far from each other, see equation (5.1)

From Equation (5.1), an acceleration due to the surface tension force can be readily introduced into the momentum Equation (3.61) by dividing it by the particle mass. The momentum equation can be rewritten as follows:

$$\frac{D\mathbf{v}}{Dt} = -\sum_{j=1}^{N} m_j \left(\frac{p_j}{\rho_j^2} + \frac{p_i}{\rho_i^2}\right) \cdot \nabla_i W_{ij} + \sum_{j=1}^{N} m_j \left(\frac{\mu_i + \mu_j}{\rho_i \rho_j}\right) \mathbf{v}_{ij} \left(\frac{1}{r_{ij}} \frac{dW_{ij}}{dr_{ij}}\right) + \frac{\mathbf{F}}{\rho_i}$$

$$+ \frac{1}{m_i} \sum_{j=1}^{N} \mathbf{F}_{ij} \tag{5.2}$$

The first advantage of the IIF method is that it is straightforward and easy to introduce. The next and the most important advantage of this approach is that it is inherently single phase; the influence from air, for example, can be ignored which reduces computational time significantly. In this method the magnitude of the surface tension depends on the strength of the force, s_{ij}. The value assigned to this strength

is not trivial and needs to be adjusted for different cases, for example, when the size of the problem changes and if the property of the fluid changes. In order to generate the appropriate surface tension for a specific fluid used an approach using droplet oscillation (see Section 5.3) will be employed.

The second method, proposed by Brackbill *et al.* [29], is based on an approach using the continuum surface force (CSF), where surface tension properties can be readily introduced into the momentum equation. The main advantage of this method is that the actual physical parameters of the given fluid is employed and therefore do not require repetitive trial and error.

The present work proposed a modified form of the CSF methodology which uses a single phase approach to the original two phase method of Brackbill *et al.* [29]. Here, the surface tension force per unit mass will be applied only to the particles on the free-surface via:

$$\mathbf{f_s} = \frac{\sigma \kappa \mathbf{n} \delta_\varepsilon}{\rho} \tag{5.3}$$

where $\mathbf{n}$ is the unit surface normal vector at the interface pointing inwards to the fluid, σ is the coefficient of surface tension which depends on the type of the fluid, κ is the surface curvature and δ_ε is the surface delta function. δ_ε is set to $1/\Delta s$ on the surface and Δs takes the value of the initial particle separation.

In the original implementation of CSF approach [122], a so called smoothed color field is used to find the surface of the fluid on which the surface tension force is applied. The smoothed color field is defined as:

$$\begin{aligned} c_i &= c(\mathbf{r}_i) \\ &= \sum_j \frac{m_j}{\rho_j} W(\mathbf{r}_i - \mathbf{r}_j, h). \end{aligned} \tag{5.4}$$

The value of the smoothed color field is dependent on a full set of neighbouring particles that the kernel in Equation (5.4) sees; particles away from the free surface will have a value approaching unity while particles on the free surface will have a value

less than 1.

Alternatively, the surface of the fluid can be determined from the divergence of particle position [81]:

$$\nabla \cdot \mathbf{r} = \sum_j \frac{m_j}{\rho_j} \mathbf{r}_{ij} \cdot \nabla_i W_{ij} \tag{5.5}$$

For 2D problems, $\nabla \cdot \mathbf{r}$ should be around 2 for the particles with a full kernel support and less than 2 for the particles close to the surface. Lee *et al.* [81] used 1.5 as a threshold to determine particles that located at the surface for the dam-breaking problem.

The unit surface normal which denotes the direction of the surface tension force is obtained from the gradient of the smoothed color field, and only calculated on surface particles using one of the above mentioned conditions:

$$\mathbf{n} = \frac{\nabla c}{|\nabla c|} \tag{5.6}$$

where

$$\nabla c = \sum_j \frac{m_j}{\rho_j} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{5.7}$$

The surface curvature can be written using the divergence of the surface normal,

$$\kappa = -\frac{\nabla \mathbf{n}}{|\mathbf{n}|} = -\frac{\nabla^2 c}{|\mathbf{n}|}, \tag{5.8}$$

Calculation of the curvature in SPH is troublesome because of the large deviations in curvature value due to the Laplacian of the color field. Adami *et al.* [1] proposed the following (which is a normalised form of the SPH divergence due to its first order consistency):

$$\kappa = -\frac{\nabla^2 c}{|\mathbf{n}|} = d\frac{\sum_j (\mathbf{n}_i - \mathbf{n}_j) V_j \nabla W_{ij}}{\sum_j |\mathbf{x}_i - \mathbf{x}_j| V_j |\nabla W_{ij}|}, \tag{5.9}$$

where d is the spatial dimension of the problem and V_j is the volume of the neighbour particles.

However, Equation (5.9) is only valid when two phase (for example, fluid and air) are

modelled. The present study introduced a new modified form for the above curvature equation for treating single phase models. It was found that ignoring the air particles, the curvature calculations from Equation (5.9) has to be halved, and further is now given by:

$$\kappa = \varepsilon \cdot d \frac{\sum_j (\mathbf{n}_i - \mathbf{n}_j) V_j \nabla W_{ij}}{\sum_j |\mathbf{x}_i - \mathbf{x}_j| V_j |\nabla W_{ij}|}, \tag{5.10}$$

where $\varepsilon = 0.5$. Validation to the validity of this revised curvature model is presented in Section 5.2.2.

The surface tension force should be applied only to particles close to the surface, and this can be insured with Equation (5.5), or alternatively, the following normal expression equation can be used:

$$|\mathbf{n}_i| \geq l \tag{5.11}$$

because $|\mathbf{n}_i|$ is zero for internal particles. To avoid numeric problems, Equation (5.11) should be applied in conjuction with Equation (5.5) to determine particles located on the surface.

5.2 Modelling Droplets

The present section aims to study the effectiveness of both the inter - particle interaction force (IIF) and continuum surface force (CSF) approaches employed to model surface forces for arbitrary droplet. Consider the case of a 2D droplet, initially defined as a square, 2.25×2.25 mm^2 in size that is made up of 2025. The fluid used is water density ($\rho = 1000$ kg/m^3, dynamic viscosity $\mu = 10^{-3}$ Pa·s). The fluid particles are uniformly distributed with particle separation, $dx = dy = 0.00005$ m. Under only the effects of surface tension forces, the initial shape of the droplet is allowed to evolve to its final equilibrium shape; the initial velocity of the all particles are stationary and each particle has the same mass.

5.2.1 IIF method

In the IIF approach, tracking of the surface of the local curvature and thus, free surface of the fluid is not necessary. The quintic kernel function (Equation (3.37)) is used as a kernel function with a constant smoothing length of $h = 1.3dx$. A constant time step, $dt = 10^{-5}$ s, is used throughout the simulation. The summation density approach is used to calculate density of the particles and the summation density is normalised (see Equation (3.48)) to improve the accuracy at the boundary particles. XSPH is applied in the simulation of the droplet to ensure that the distribution of the particles are homogeneous.

The result shows that the droplet oscillates under the effect of the surface tension force, as shown in the Figure 5.2. Various modes of oscillation is seen as the oscillating droplet evolves from a square shape configuration to a diamond one and back. After about 3 oscillations, the droplet oscillation damps and reaches its final circle shape without any sharp corners. Figure 5.2 clearly shows that the size of the final droplet is in the same order as its initial shape. But, the configuration of particles within the circular final droplet shape contains unphysical rings. This unphysical rings appear even with initial circle droplet shape as shown in Figure 5.3.

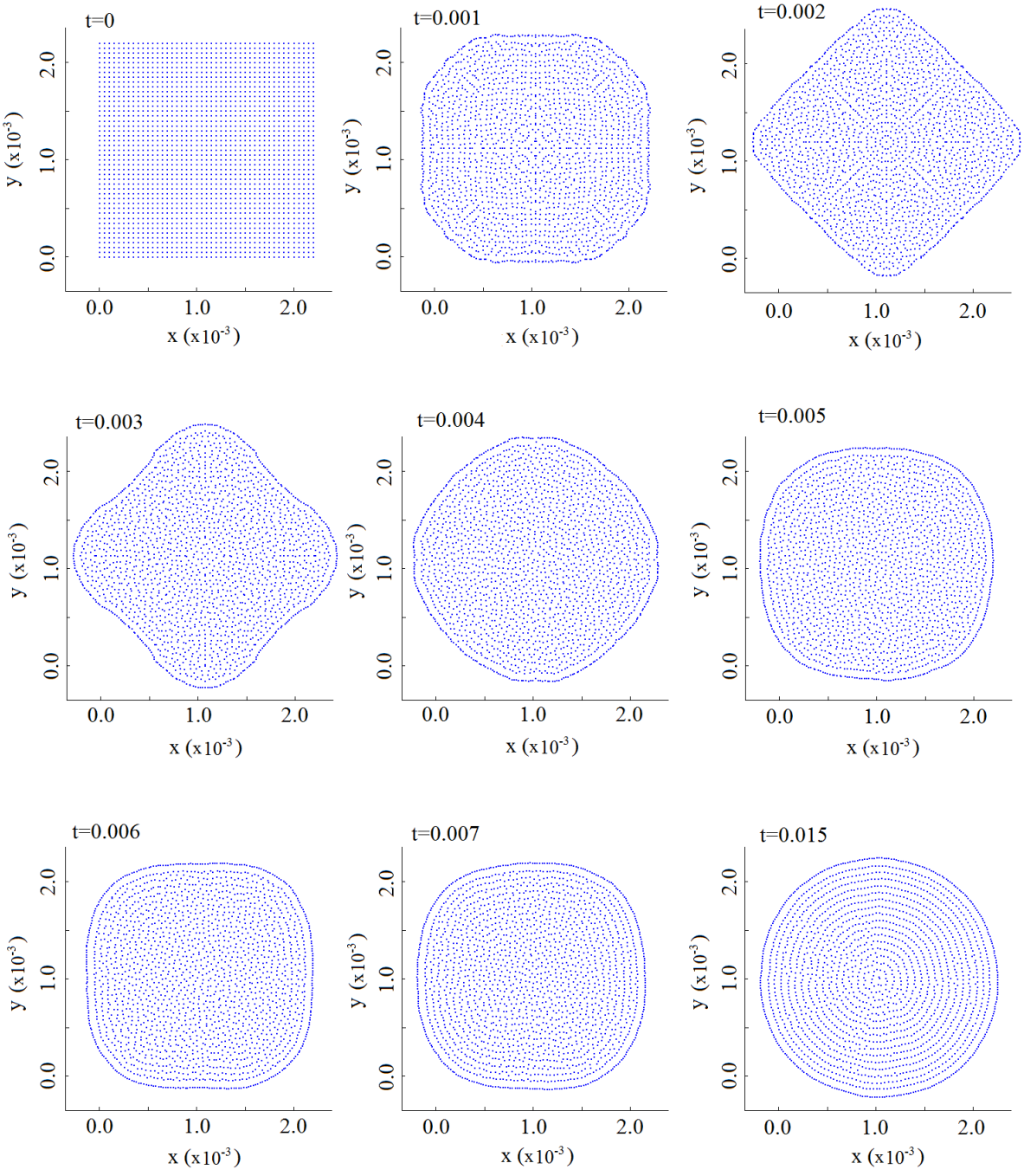


FIGURE 5.2: Formation of a droplet from initial square shape using the IIF approach

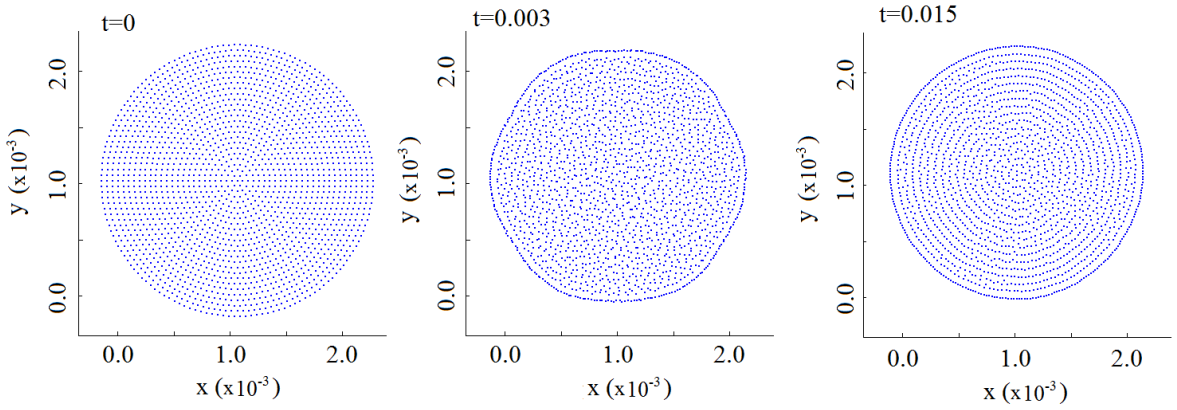


FIGURE 5.3: Formation of a droplet from initial circle shape using the IIF approach

This unphysical rings can be resolved using a few methods. First, a quadratic kernel normally employed for high velocity impact (HVI) problems developed by Johnson *et al.* [77] is introduced. The quadratic kernel function and its first derivative are given as written below, respectively:

$$W(\mathbf{r}-\mathbf{r}',h) = W(q,h) = \alpha_d \begin{cases} \frac{3}{8}q^2 - \frac{3}{2}q + \frac{3}{2} & 0 \le q \le 2 \\ 0 & q > 2, \end{cases} \tag{5.12}$$

$$\frac{dW}{dr}(q,h) = \frac{\alpha_d}{h} \begin{cases} \frac{3}{4}q - \frac{3}{2} & 0 \le q \le 2 \\ 0 & q > 2, \end{cases} \tag{5.13}$$

where $\alpha_d = \dfrac{1}{\pi h^2}$ for two dimensional space.

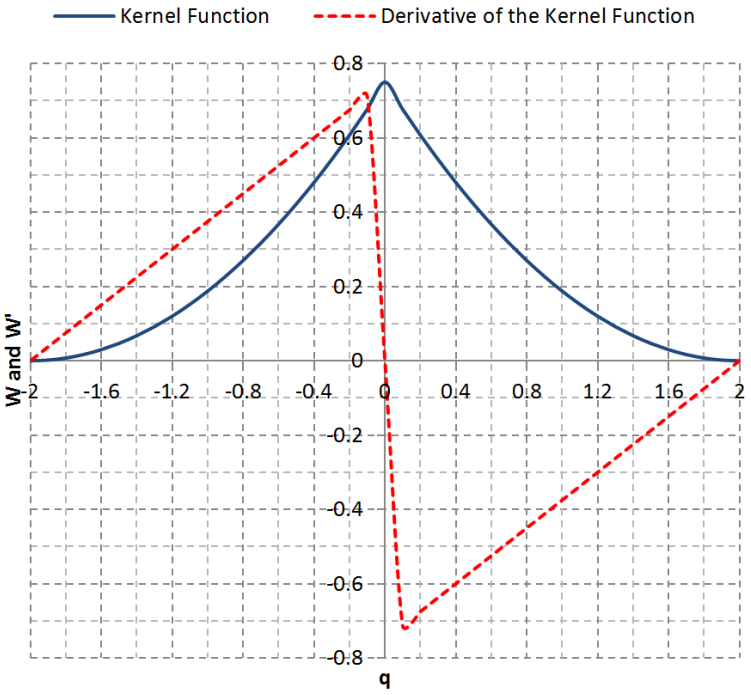


FIGURE 5.4: The quadratic kernel function and its first derivative. The derivative of the quadratic kernel function constantly increases as the SPH particles come closer and prevents particle clusterings when this kernel is used only in calculation of the momentum due to the pressure term.

As it can be seen from Figure 5.4 the derivative of the quadratic kernel function increases in magnitude almost constantly as particles come closer, for this reason, this kernel is used only for calculation of the momentum equation due to the pressure term to prevent particle clusterings and therefore solves the unphysical ring problems shown in Figure 5.5. However, the particle separations of the surface particles are not resolved and are close to each other.

The second method for solving the unphysical ringing problem is by applying a repulsive force (see Equation (3.76)) between all fluid particles in the Lennard-Jones form, usually applied between fluid and solid particles to prevent fluid-solid particles penetration. This force is considered as an external force and implemented only in the momentum equation. As a result, particle distribution is more homogeneous as it shown in Figure 5.6.

As a conclusion, applying the IIF method will lead to the particle clustering. In general, introducing the quadratic kernel in the pressure term and introducing the Lennard-Jones force as an external force helps to avoid particle clustering and they do not affect to the dynamics of the droplet oscillation as it seen from the Figures 5.2, 5.5 and 5.6. However, the surface of the droplet at equilibrium is not an ideally smooth circle for all cases. Another limitation or unphysical property of this method is that the strength of the surface tension force does not depend on the geometry of the surface. For example, the strength of the surface tension force is the same at the surface of the initial square shape even at the sharp corners as shown in Figure 5.7a. The vector plots from Figures 5.7 showed that the surface tension force is near zero at the internal fluid particles. Figure 5.7c further illustrates that the surface tension force for the internal particles is significantly smaller than the force applied on the surface but is random in direction caused by numerical noise; this may have an adverse effect to the long time scale stability of the droplet.

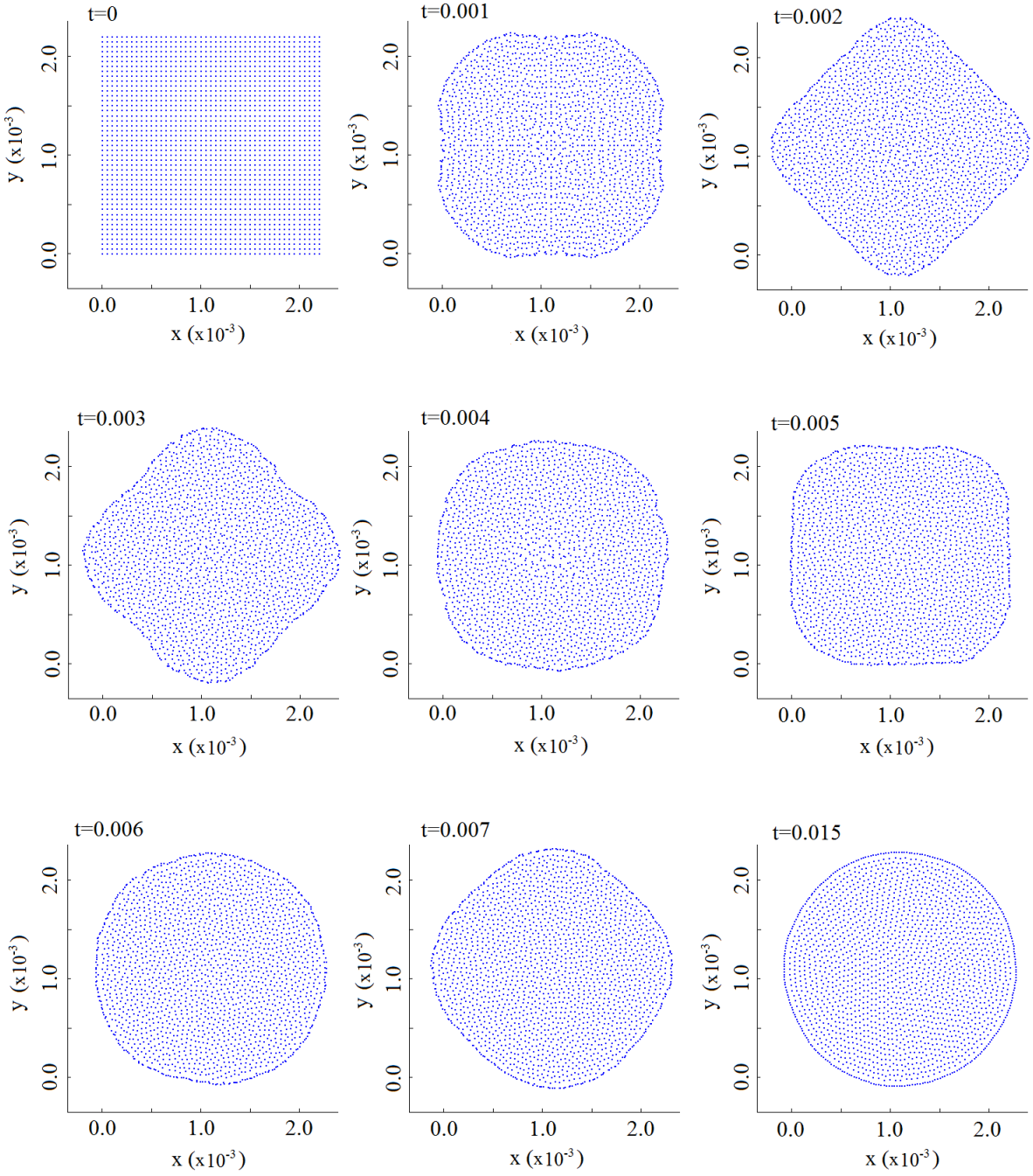


FIGURE 5.5: Formation of a droplet from initial square shape using the IIF approach with additional quadratic kernel to solve unphysical rings problem

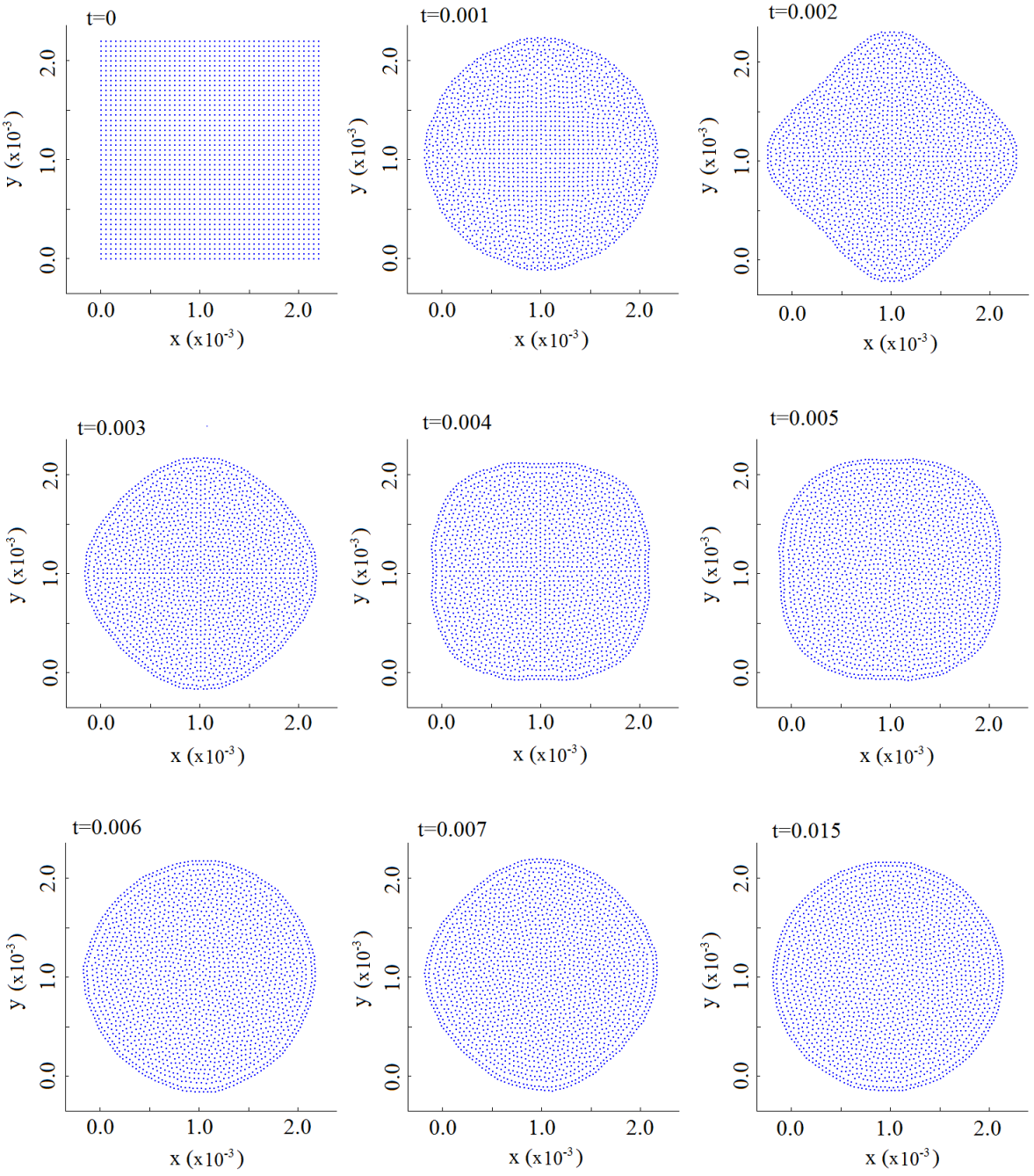


FIGURE 5.6: Formation of a droplet from initial square shape using the IIF approach with additional repulsive force in the Lennard-Jones form to solve unphysical rings problem

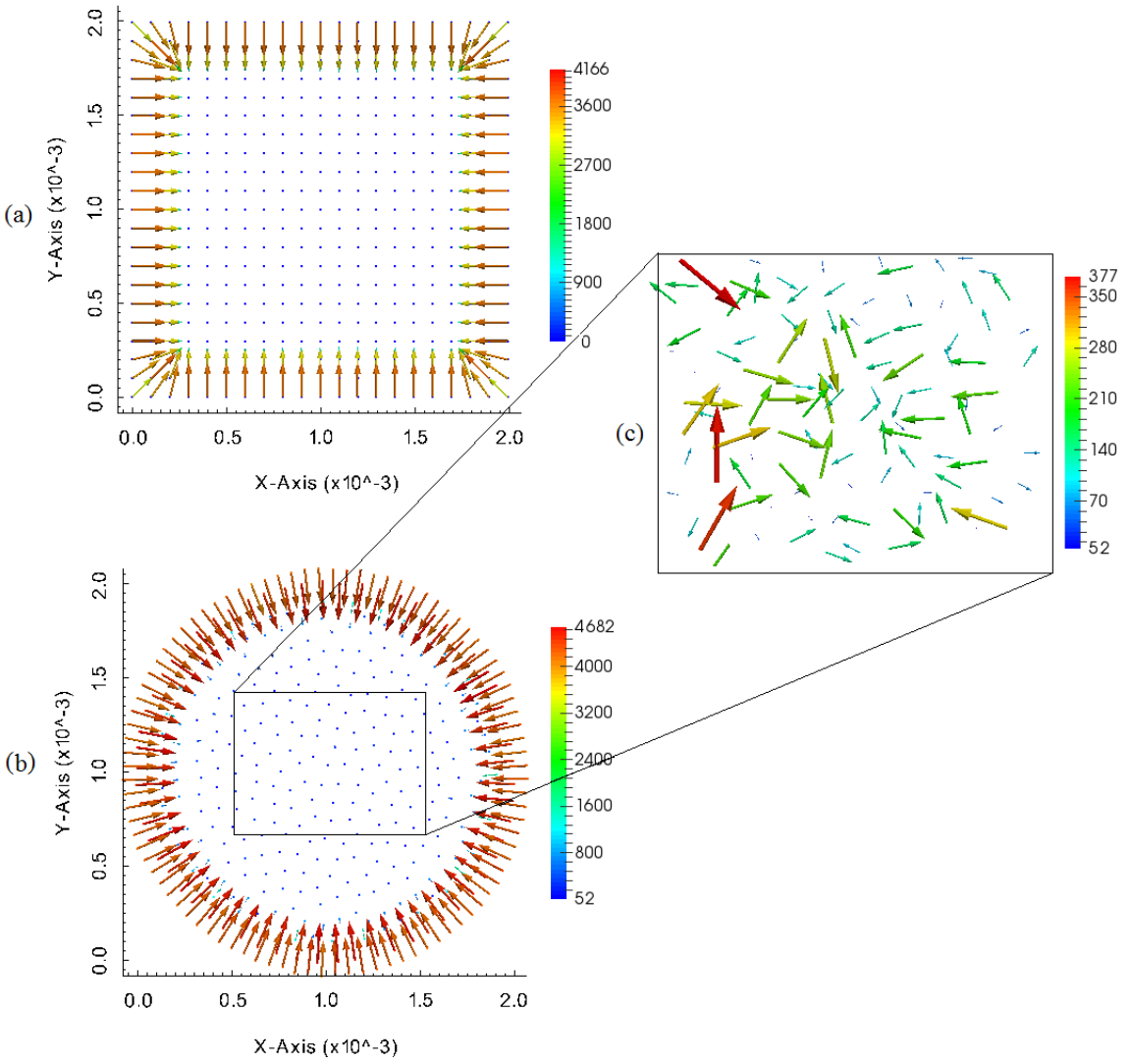


FIGURE 5.7: The surface tension force vector; a) on the surface of the initial square shape, b) on the surface of the equilibrium circle shape, c) at the internal particles of the droplet. Color bars correspond to the strength of the surface tension force.

5.2.2 CSF method

In this section, a 2D droplet similar to the one employed in Section 5.2.1 is modelled using the CSF method. The continuity density approach (Equation (3.50)) and the cubic spline kernel (Equation (3.33)) was used.

The most important challenge in the simulation of the droplet under the effect of surface tension force with the CSF approach lies in the computation of curvature (see Equation (5.10)) and its unit normal vector (see Equation (5.6)) of the surface. Both

the curvature and unit normal vector on the droplet surface are studied to validate the CSF method for surface tension force. The curvature and unit normal vector are highly dependant on particle distribution. For this reason, the surface tension force is not taken in account to validate the proposed curvature approach and the unit normal vectors direction. A circular droplet shape with equally spaced particles will be used to provide a smooth and consistent particle distribution.

Here, a two-dimensional water droplet of radius 1 mm is considered to verify the proposed curvature equation (see Equation (5.10)) for the single phase problem with initial particle spacing set at 0.1 mm, 0.05 mm and 0.025 mm intervals, giving particle resolutions of 331, 1261 and 4921, respectively, as shown in Figure 5.8. The curvature is calculated only for the surface particles. Results from the simulations were compared against the analytical solution

$$\kappa = \frac{1}{R} \tag{5.14}$$

where R is the radius of the droplet.

The results are in good agreement with an error of 0.4%. And this proposed curvature formulation (Equation (5.10)) does not depend on the particle resolution as it can be seen from the common color bar in Figure 5.8. The unit norm vector is also calculated only for the surface particles. Directions of the unit norm vector is expected to point inward towards the centre of the droplet because of the initial shape of the droplet. Figure 5.9 shows the directions of the unit normal vector with particle resolutions of 331, 1261 and 4921, respectively. They all showed its direction pointing inwards and to the centre; they do not depend on the resolution.

(a) 331 particles

(b) 1261 particles

(c) 4921 particles

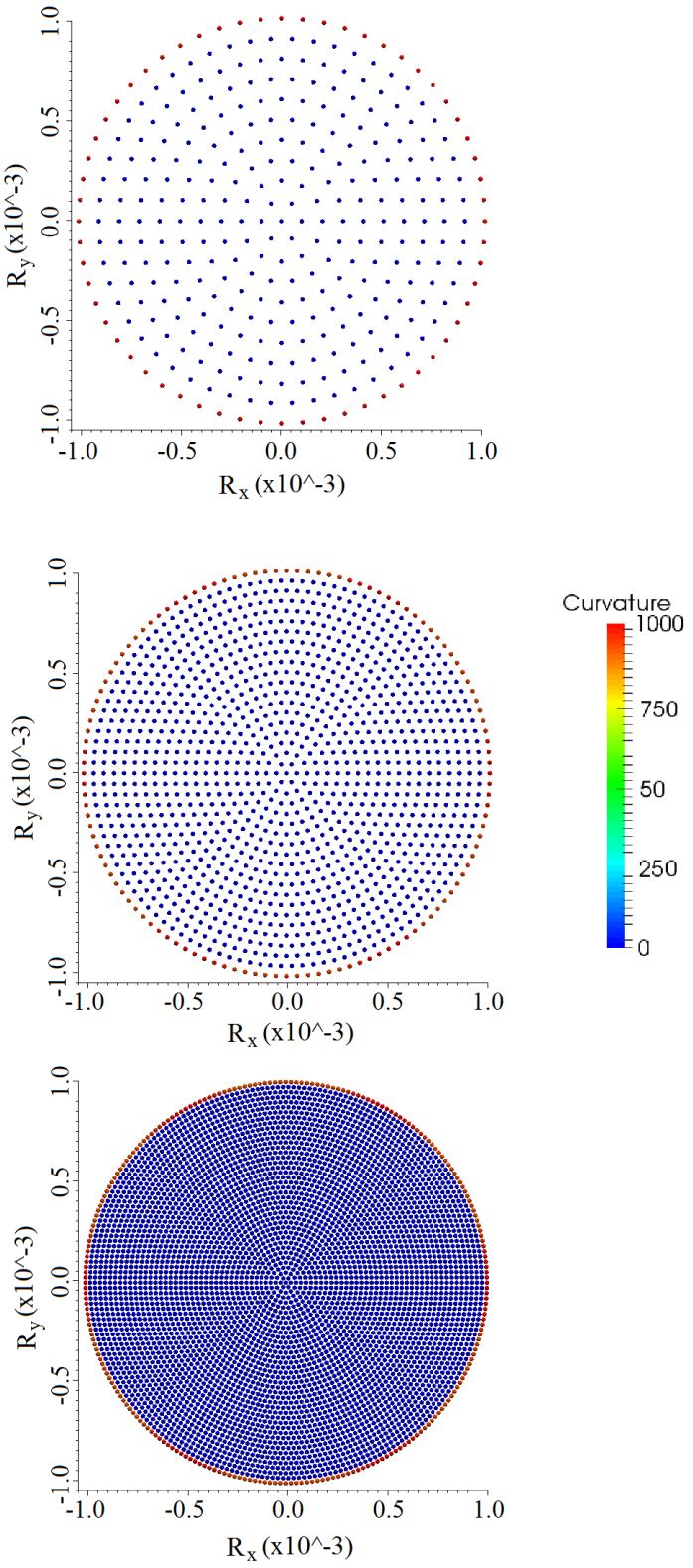


FIGURE 5.8: The curvature of 2D droplet surface with radius of 1mm with particle resolutions of: a) 331, b) 1261 and c) 4921

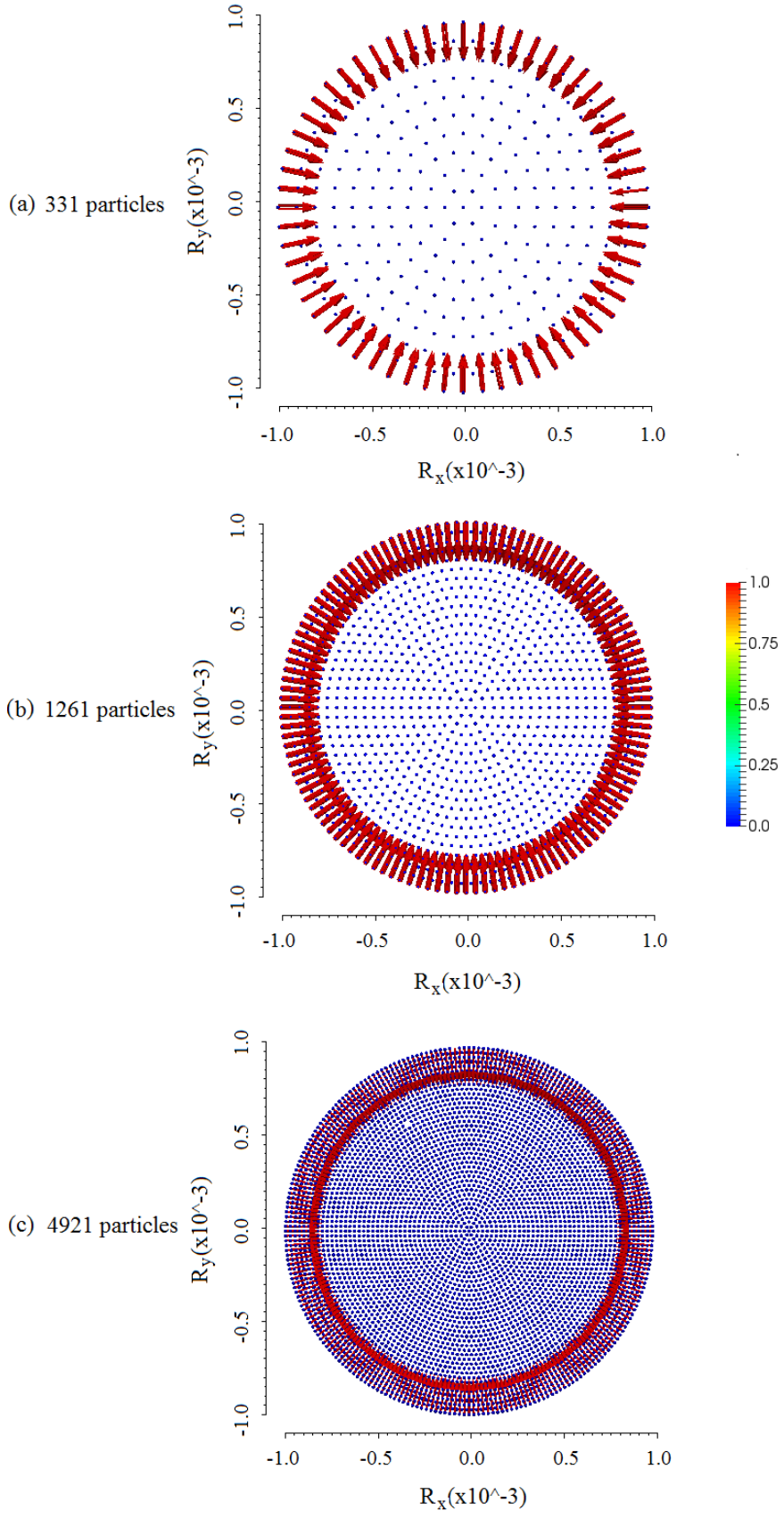
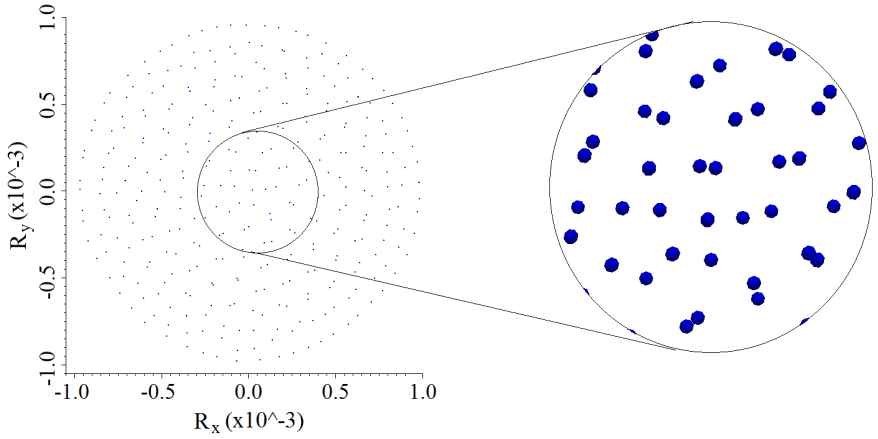
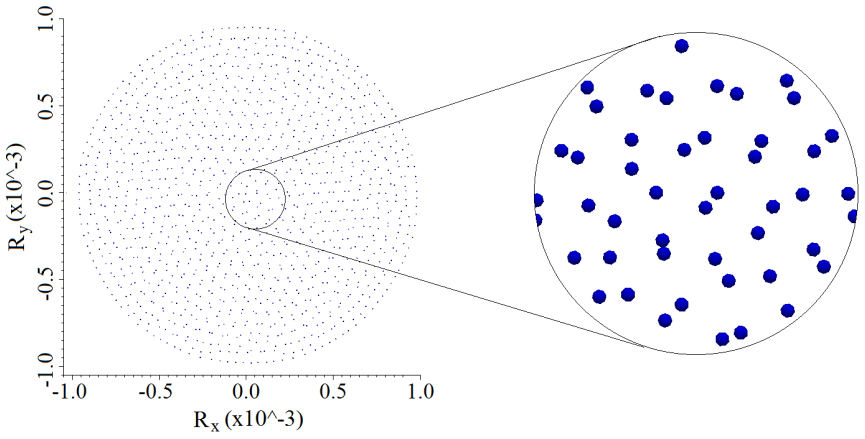


FIGURE 5.9: 2D droplet with the normal vector on the surface.

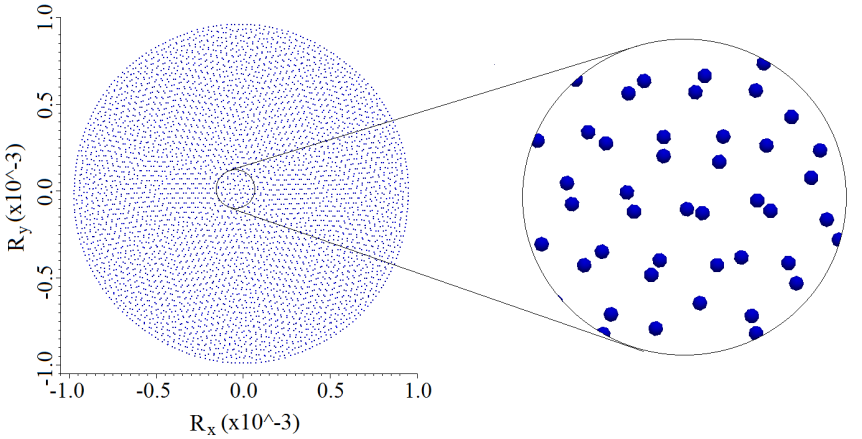
A surface tension force is applied to the surface particles of the droplet with radius of 1 mm after ensuring that the curvature and the unit normal vector have the correct value and direction. After applying the surface tension force, the water particles re-equilibrates to their stable position. However, particle clusterings/clumpings are observed and the particle distributions are not ordered as shown in Figure 5.10. The CSF method requires very ordered particle distribution since the curvature highly depends on particle distribution. Any unexpected and unphysical particle distributions may lead in increase/decrease of the curvature which increases/decreases the surface tension force. Replacing the cubic kernel with the Wendland kernel (Equation 3.39) makes the particle distribution very ordered as shown in Figure 5.11. However, the Wendland kernel can not always guranttee ordered particle distribution for complex movements, for example, when initial droplet shape is started from square, some particle clusterings and sharp edges appear at the corners as shown in Figure 5.12 and additional repulsive forces such as the in the Lennard-Jones potential may be required.



(a) 331 particles

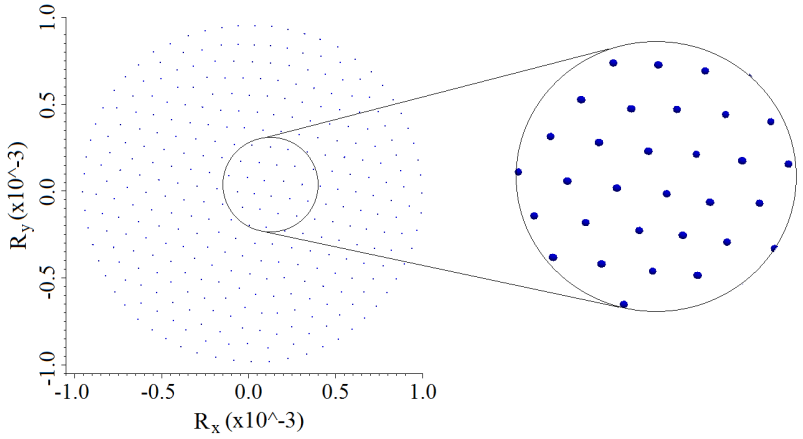


(b) 1261 particles

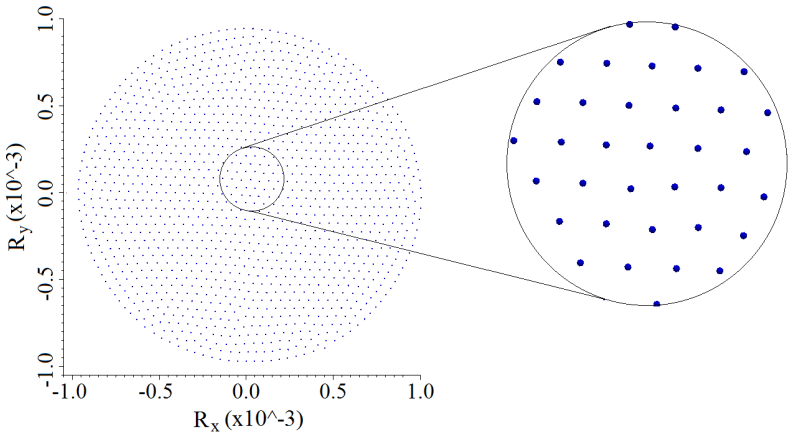


(c) 4921 particles

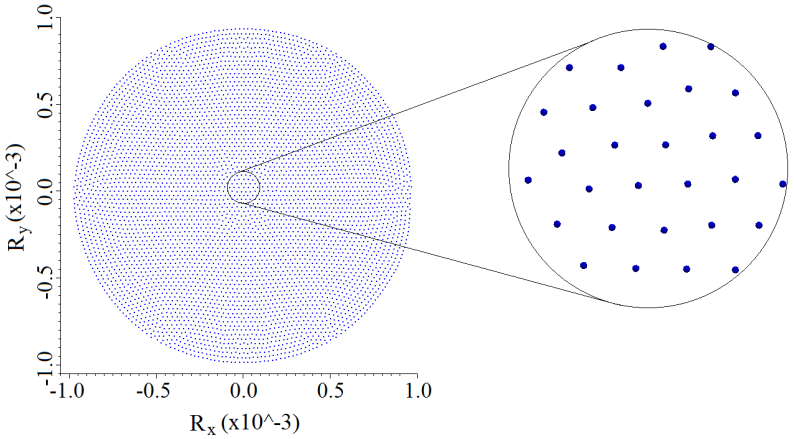
FIGURE 5.10: Particle clusterings and clumpings occur with the cubic spline kernel for the particle resolutions of: a) 331, b) 1261 and c) 4921.



(a) 331 particles



(b) 1261 particles



(c) 4921 particles

FIGURE 5.11: Droplet simulation with Wendland kernel for the particle resolutions of: a) 331, b) 1261 and c) 4921.

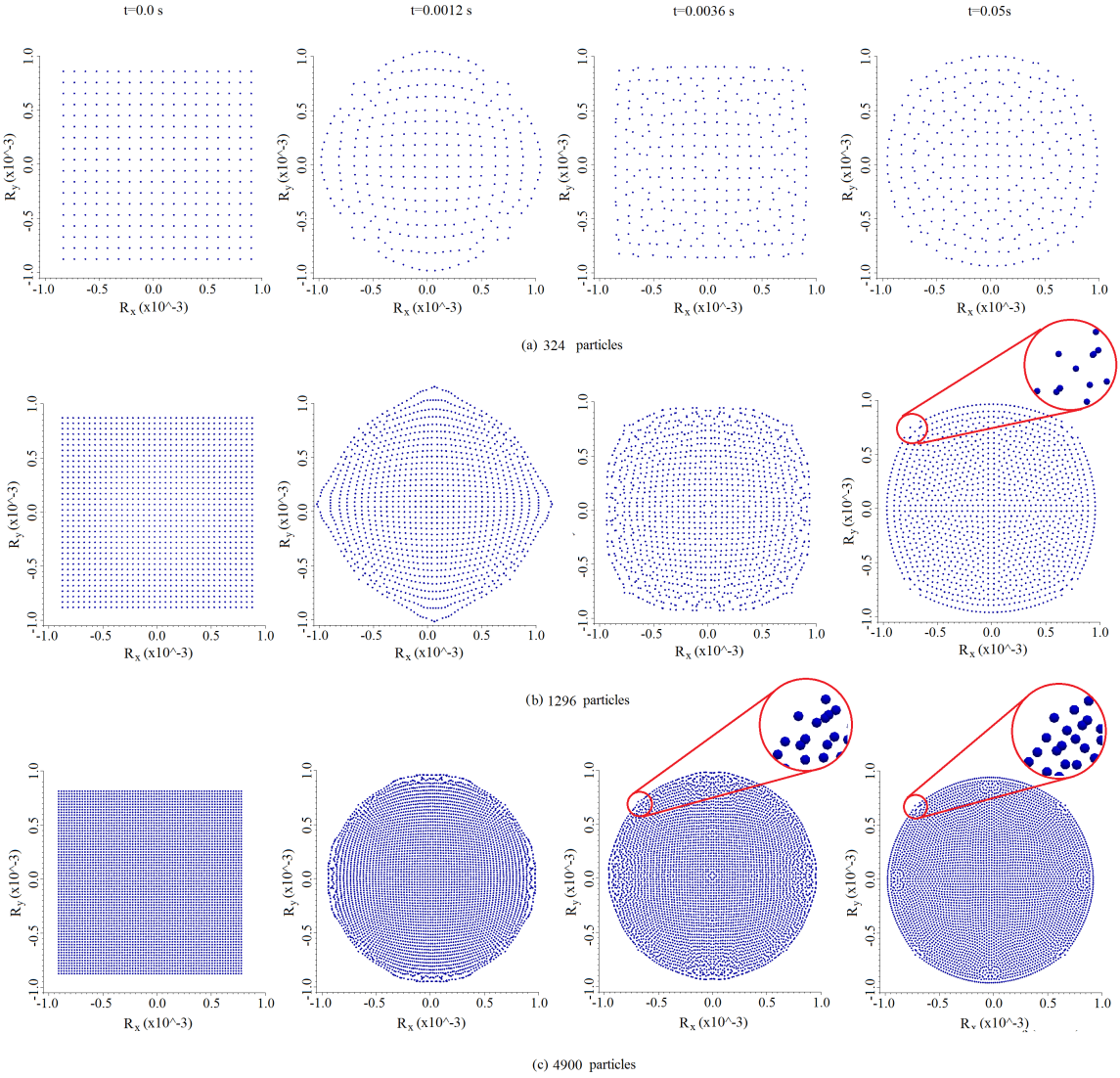


FIGURE 5.12: Droplet simulation with Wendland kernel for the particle resolutions of: a) 324, b) 1296 and c) 4900. Red circles show particle clusterings when zoomed in.

5.3 Droplet Oscillation

5.3.1 IIF method

The effect of the surface tension force is investigated by considering the case of an oscillating droplet. In the IIF approach, the first task is to determine of the force parameter, s_{ij}, that corresponds to the surface tension, σ, of the fluid. And this can

be performed by comparing the oscillation period of the droplet with the analytical period [128]:

$$\tau = 2\pi\sqrt{\frac{R^3\rho}{6\sigma}} \tag{5.15}$$

where ρ is the density of the droplet, R is the equilibrium radius of the droplet and σ is the surface tension coefficient of the droplet.

The above is performed via trial and error where the strength of the force is adjusted according to the fluid property.

As an example, a 2D droplet was first generated with initial shape of square, later the IIF (Equation (5.1)) was applied to the all SPH particles and the simulation was allowed to run until it reached its equilibrium droplet shape in the absence of gravity. The initial conditions of the droplet are as shown in Table 5.1.

	Fluid
Density, ρ (kg/m^3)	1000
Pressure, p (Pa)	0
Velocity, $\vec{v}$ (m/s)	0
Acceleration, $\dot{\vec{v}}$ (m/s^2)	0
Initial particle separation, $dx = dy$ (m)	8.888×10^{-5}
Smoothing length, h (m)	$1.3dx$
Dynamic viscosity, μ $(Pa \cdot s)$	10^{-3}

TABLE 5.1: The initial conditions of the SPH particles for 2D water droplet simulation

The final equilibrium circle shape of the droplet was then deformed into an oval shape with the following expression:

$$\begin{pmatrix} x^* \\ y^* \end{pmatrix} = \sqrt{\frac{2}{\sin\theta}} \begin{bmatrix} x\sin(\theta/2) \\ y\cos(\theta/2) \end{bmatrix} \tag{5.16}$$

where $\theta = \varepsilon\pi$, the eccentricity $\varepsilon = 0.55$, x, y and x^*, y^* are the components of position of the each particle before and after deformation. The deformed oval droplet is then

let go to oscillate.

Figure 5.13 shows the oscillation snapshots at different time, t. In this study, the quadratic kernel (Equation (5.12)) was used to solve clustering problems and to achieve more uniform distribution of the SPH particles. The analytical period of the oscillation of a 2D water droplet with a radius of $1.6 \cdot 10^{-3}$ m according to the Equation (5.15) is equal to 0.019 s. A few oscillations were performed to determine out the correct corresponding strength of the force for a water droplet with the same radius. Figure 5.14 shows the change of the radius in the x and y directions against time with the strength of the force $s_{ff} = 0.008$ and it is clear that the period of the oscillation for this given size of droplet is equal to 0.019 s. As it can be seen from Figure 5.14, oscillation of the droplet damps out quickly and reaches steady state in only 4 oscillations due to the non-zero surface tension force at internal particles (see Section 5.2.1). However, Apfel *et al.* [7] recorded 82 oscillation for a water drop during an experiment in near zero gravity in space. From here, concluded that the droplet modelled with the IIF method behaved more viscously than it is supposed to be (water). Figure 5.15 shows that the strength of the force $s_{ff} = 0.008$ remains constant for different particle resolutions in order to satisfy the required fluid property (water). However, the damping rate increases with increasing the particle resolution.

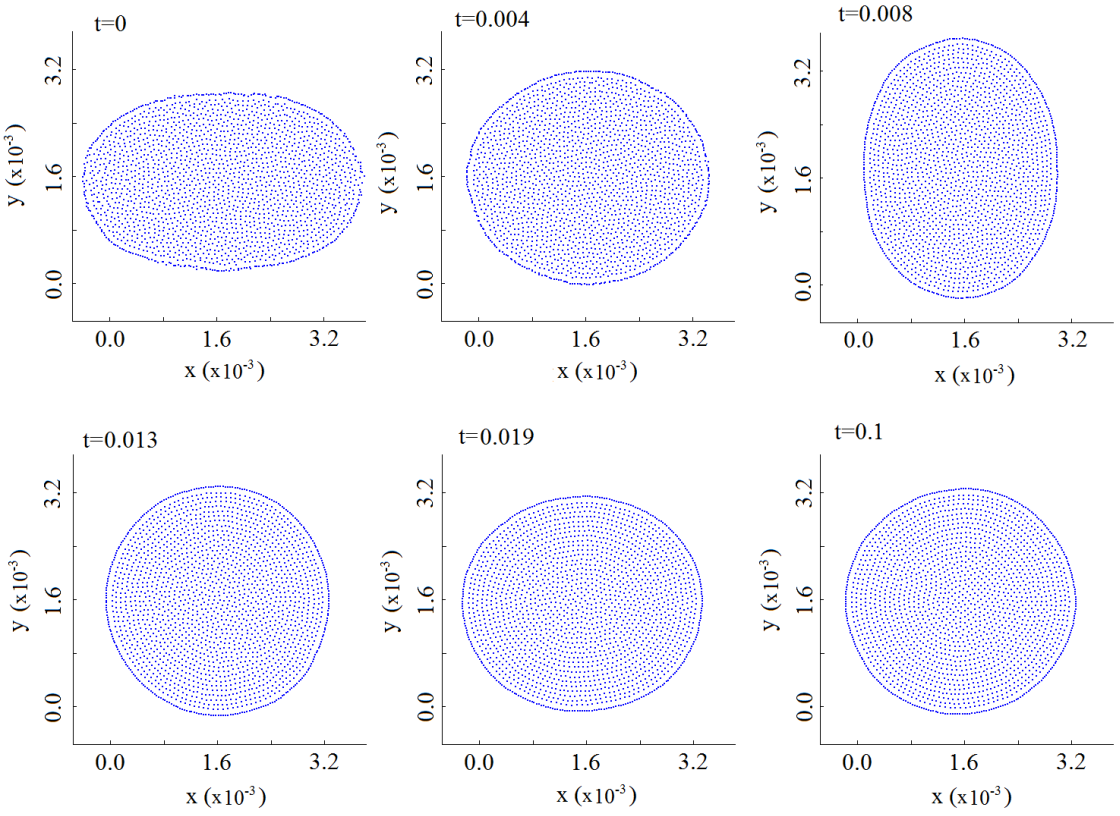


FIGURE 5.13: Snapshots of the oscillation of a 2D droplet with an initial oval shape

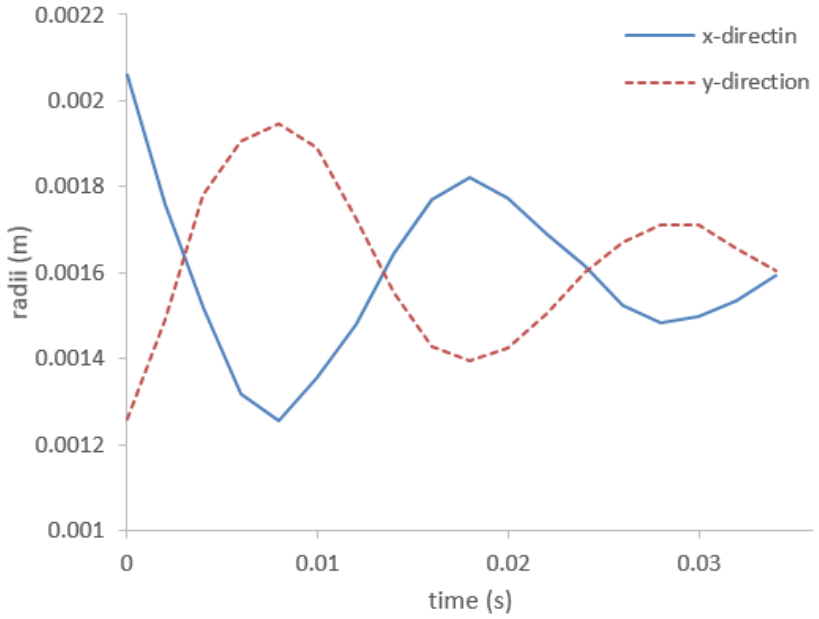


FIGURE 5.14: Oscillation of a 2D droplet with a quadratic kernel function

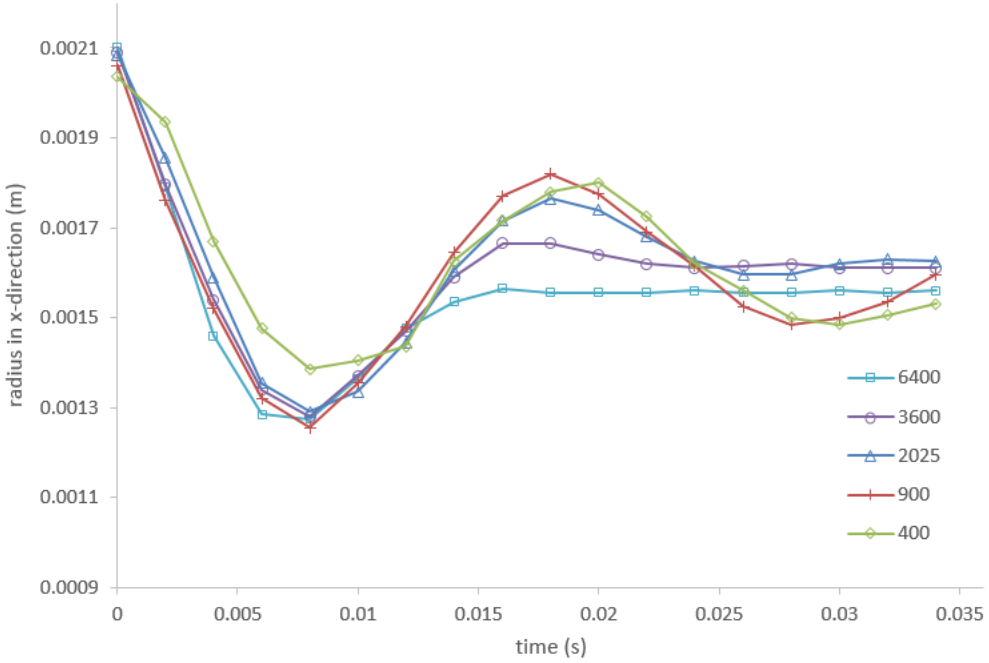


FIGURE 5.15: Oscillation of a 2D droplet with a quadratic kernel function with different particle resolutions

5.3.2 CSF method

2D water droplet is considered presently using the Wendland kernel. A continuity density approach is used to update the density field with a reference density, $\rho_0 = 1000$ kg/m^3, and a surface tension value, $\sigma = 0.072$ N/m, which corresponds to the water is set. The physical properties of the water droplet is verified by oscillating the droplet and comparing the period of the oscillation given in Equation (5.15) The oscillation of the droplet is started from an oval initial shape by applying the Equation (5.16) [128] to the equilibrium droplet shape.

Figure 5.16 shows the evolution of the droplet radius with time taken along the cross-section in the x-direction for both the IIF and CSF method. Both the IIF and CSF methods yields the analytical period value of 0.019 s for water. The strength of the surface tension force s_{ff} that corresponds to the water was achieved after a few simulation attempts for the IIF method. For the CSF method, the real surface tension coefficient of the water was used and the period of the oscillation from the simulation

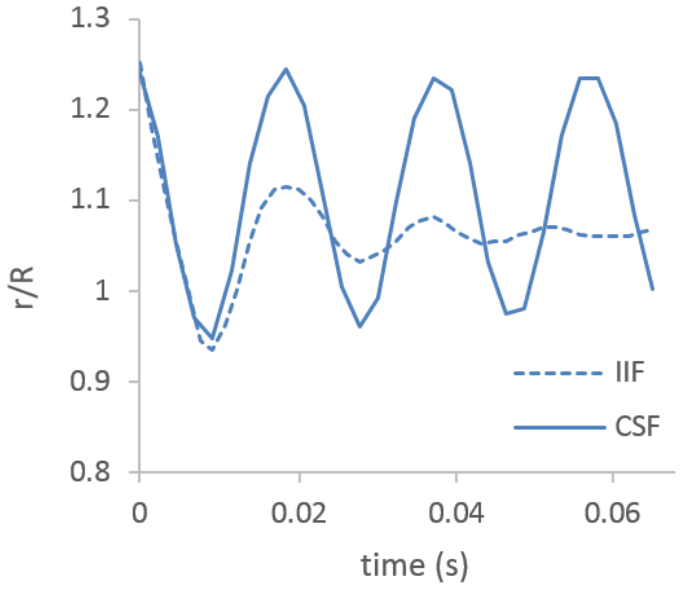


FIGURE 5.16: Time evolution of the radius displacement in x-direction of an oscillating droplet modelled using the IIF and modified CSF methods.

agrees with the analytical period. However, the results showed that the IIF approach damps very quickly and reaches steady state in only 4 oscillations. This is due to the ever-present dissipating internal inter-particle forces within the droplet. On the contrary, the droplet modelled using the proposed CSF approach oscillates far longer and damps out in approximately 60 oscillations. Comparisons with experimental results in Apfel *et al.* [7] reported that a squeezed droplet of 6.6 cm^3 in micro-gravity oscillates as much as 82 times. The results obtained from the modified CSF approach was performed in two-dimensions and one would expect that the surface tension contribution would be smaller and thus the number of oscillations will be fewer compared to three-dimensional ones. Figure 5.17 shows the droplet oscillation snapshots taken at fixed time intervals to compare droplet evolution and particle distribution of the CSF and IIF methodologies. It clearly shows the distinctive particle arrangements between both approaches. The former has a distinctive particle layer at the free-surface boundary interface. This is due to internal inter-particle forces and has been similarly observed in Nugent and Posch [128]. The latter CSF approach showed a much cleaner and evenly particle distribution. The use of the Wendland kernel contributed to minimising the effect of tensile instability [98] while the surface tension forces act only at the interface boundaries. Small voids can be seen for these particle

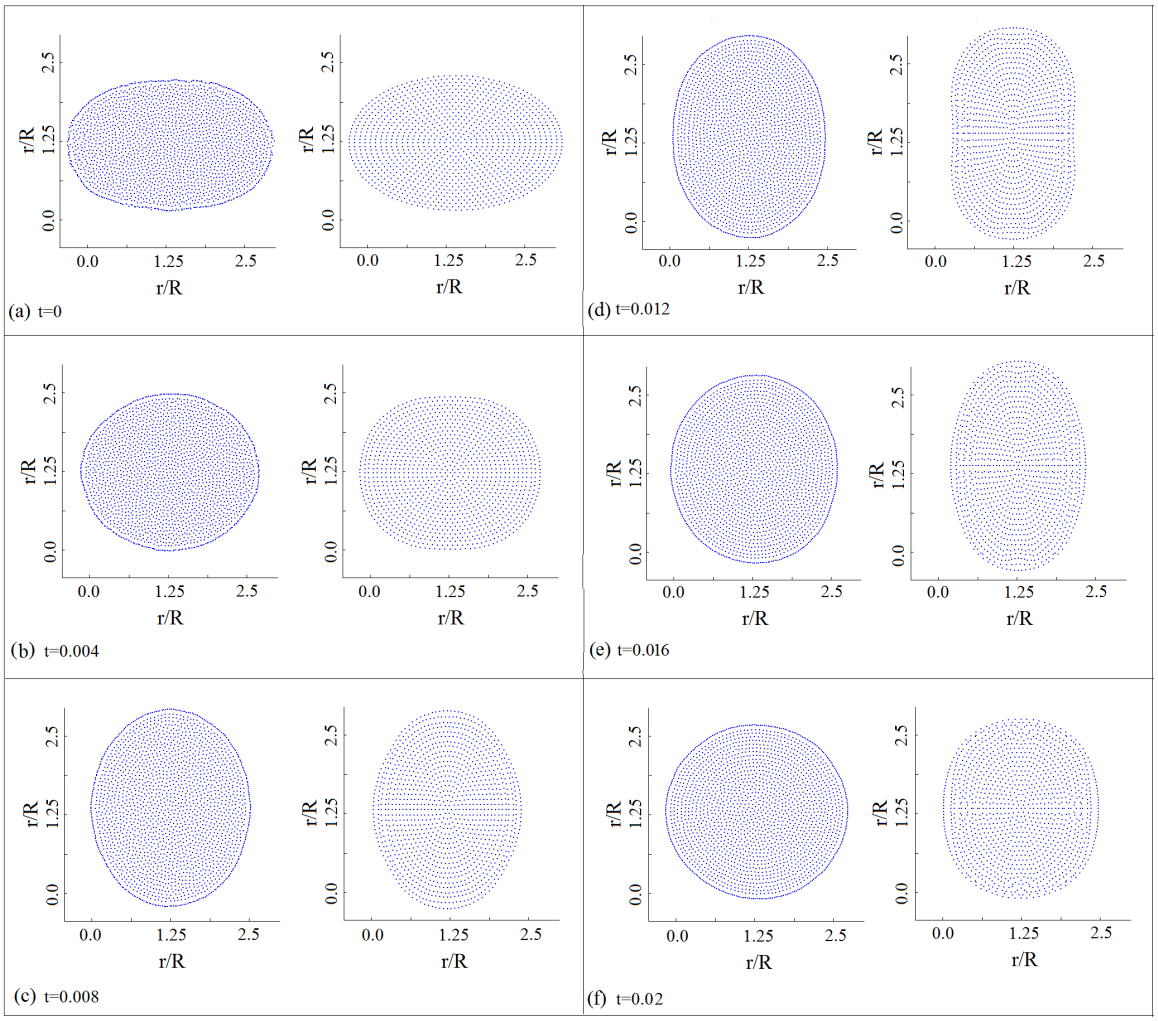


FIGURE 5.17: Oscillation snapshots of droplet using the IIF and modified CSF methods taken at various time intervals using a resolution of 1261 particles.

resolution, but these disappear when high particle resolution are used (see Figure 5.20). In addition, droplets at higher resolution tend to hold their shape better but this does not mean that they oscillate larger in amplitude and longer in time. Figure 5.18 shows the radial displacement evolution of the oscillating droplet at different resolution. It shows that as the resolution increase, the amplitude of the oscillation decays much faster and resulted in slightly smaller number of total oscillations, inline with what one would expect from a more tightly packed distribution. The effect of using the weakly incompressible assumption is supported by analysing the changes in density over the first period of oscillation where changes in density is largest. This is shown in Figure 5.19 that the maximum error in density variation is worst at $t = 0.015$ s giving a maximum error value of 0.6%. Nonetheless, the overall density

profile is relatively smooth and this smooth variation persist throughout the remaining oscillations. The results obtained are consistent and accurate within acceptable limits with those of the weakly compressible assumption given in Monaghan [117].

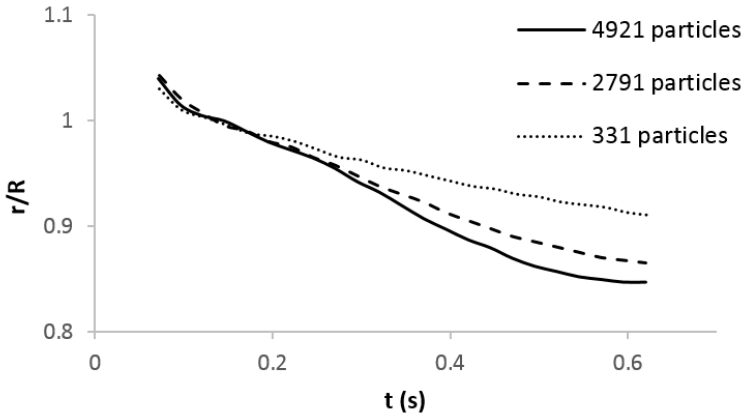


FIGURE 5.18: Evolution of the radial oscillating droplet at various resolutions using the modified CSF approach.

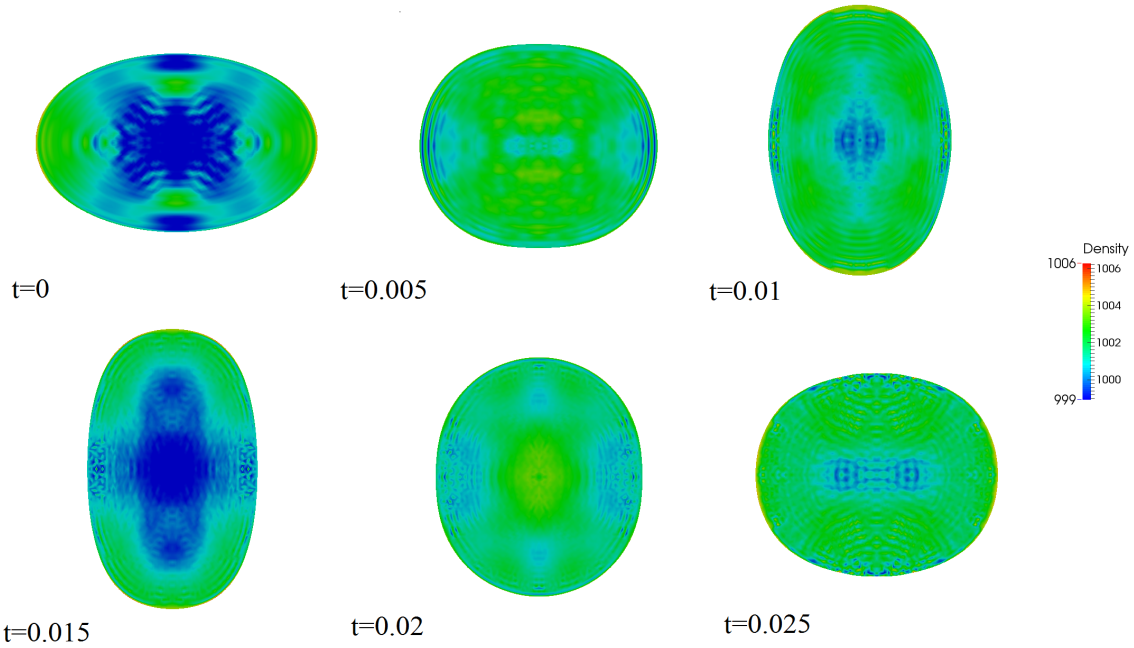


FIGURE 5.19: Density distribution of the modified CSF approach taken at different snapshots in time at the first period of oscillation.

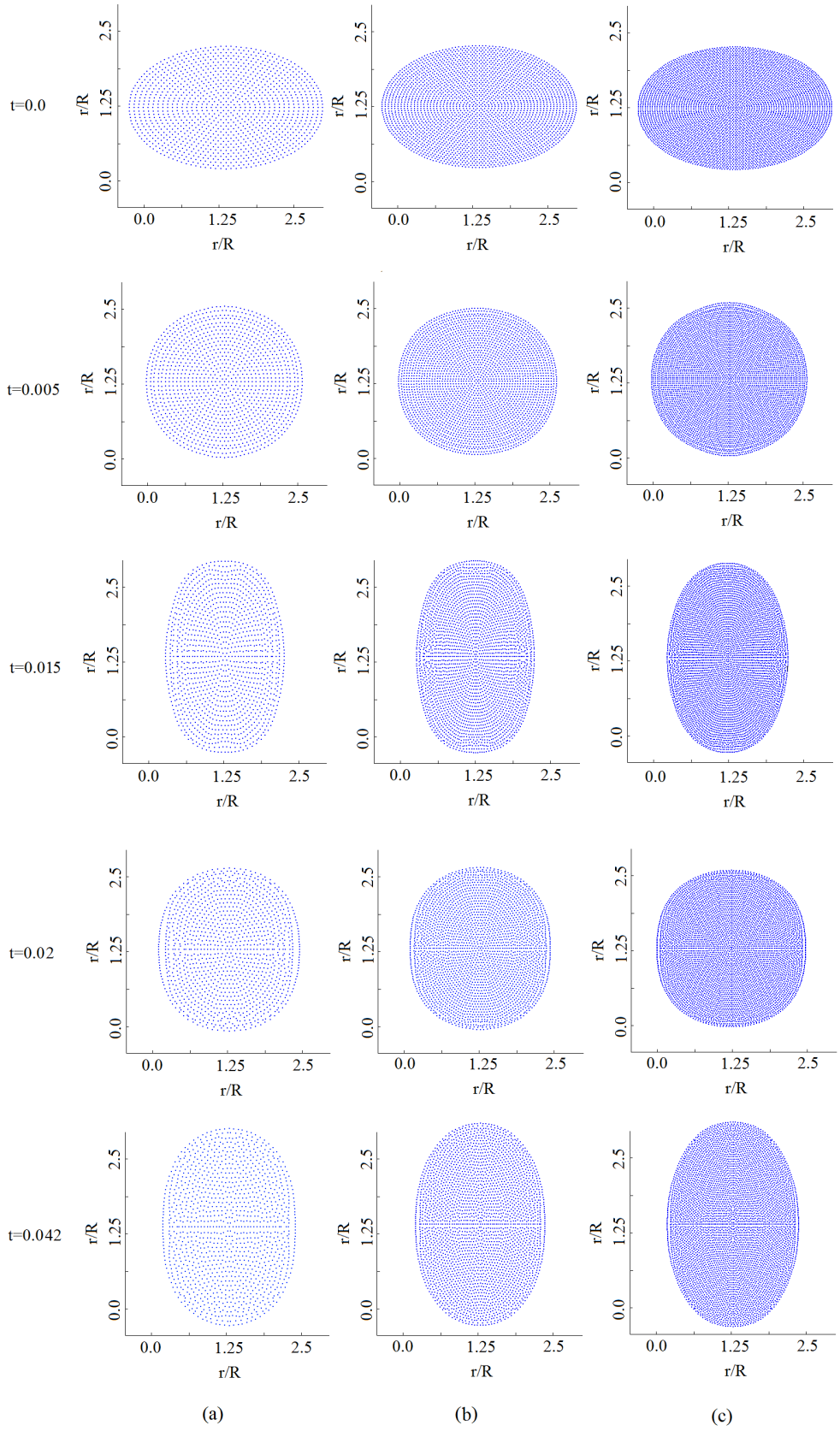


FIGURE 5.20: Droplet oscillation snapshots using the modified CSF method taken at various times with particle resolutions of (a) 1261, (b) 2791, and (c) 4921.

Chapter 6

Droplet Spreading

6.1 Static Equilibrium Contact Angle

The present section aims to investigate surface tension effects on droplet spreading. The test cases examined in subsections below uses three different approaches, namely the IIF, CLF and disjoining pressure methods to control static equilibrium contact angles during droplet spreading.

6.1.1 IIF method

Tartakovsky [146] and Zhou [164] successfully studied the wetting effect with the IIF method for two-phase problems which makes the solver more expensive in terms of computing time. Tartkovsky [146] is used a complicated Van der Waals equation of state to update the pressure where it requires the use of three control variables making it not a straightforward approach. Here, the IIF method is applied to the single-phase problem with a simple Tait's equation of state to update the pressure term which depends on the density change. The IIF method is straightforward and simple to code.

Consider the droplet spreading case presented in Figure 6.1 that shows the initial setup of the fluid laying on the substrate on which it is going to spread. The simulation begins with an initial square droplet shape. Due to the present of solid substrate, the strength of the force between the fluid and solid particles is introduced, s_{sf}, which controls the static equilibrium contact angle. Fluid particles that are adjacent to the solid substrate will experience this additional force and balances with the fluid and fluid interaction forces, s_{ff}, at the moving interfaces to attain its equilibrium state. The strength of force between the fluid and fluid particles were set to $s_{ff} = 0.008$ which corresponds to the water property for this given volume of droplet (see Subsection 5.3.1) while that between the fluid and solid is tuned within the range of $0 \leq s_{sf} \leq 0.008$ in order to achieve the desired static equilibrium contact angle. The substrate is modelled using three layers of dummy particles with the same initial parameters for water presented in Table 6.1. A cubic kernel is used with smoothing length of $h = 1.3dx$ where dx is a initial separation of the particles. A reference sound speed of 50 times the maximum velocity, $v_{max} = \sqrt{2gR}$, is chosen where R is the radius of the droplet and g is the gravity. Continuity density approach is used to update the density field with a reference density, $\rho_0 = 1000$ kg/m^3. The repulsive force in the Lennard-Jones form (given in Equation 3.76) is used to prevent unphysical clusterings of particles (see Section 5.2.1).

Figure 6.2 shows the resultant equilibrium stationary shape at which the static equilibrium contact angle measured from the interaction forces between the solid and fluid particles, s_{sf}, for the fluid particle resolutions of 528. It shows that the IIF method, by controlling s_{sf} for fluid water properties, successfully models the spreading of a droplet on a substrate. However, as noted in Figure 6.2q and 6.2r, the steady state droplet shape is not perfectly symmetric at smaller static contact angles. This is due to the low particle resolution and in order to improve this limitation, higher particle resolutions were considered.

In order to carry out the convergence study of the problem, different particle resolutions were used. Figures 6.3 and 6.4 show the evolution of static contact angle, θ, and its dependence on the interaction force between the solid and fluid, s_{sf}, for fluid

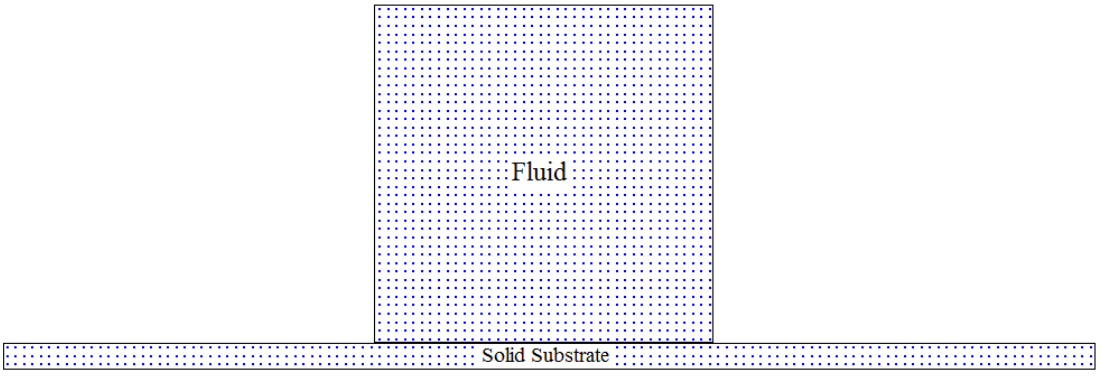


FIGURE 6.1: Initial shape of the fluid before applying the surface tension force.

	Fluid
Density, ρ (kg/m^3)	1000
Pressure, p (Pa)	0
Velocity, $\vec{v}$ (m/s)	0
Mass, m (kg)	$1.4 \cdot 10^{-6}$
Acceleration, $\dot{\vec{v}}$ (m/s^2)	0
Initial particle separation, $dx = dy$ (m)	$3.75 \cdot 10^{-5}$
Smoothing length, h (m)	$1.3dx$
Dynamic viscosity, μ $(Pa \cdot s)$	10^{-3}

TABLE 6.1: Initial parameters of the droplet.

particle resolution of 900 and 1940, respectively, by keeping the same droplet size. To give a clearer picture, Figure 6.5 shows the mean static equilibrium contact angle (θ) for different interaction force between the fluid and solid (s_{sf}) and its standard deviation due to the effects of varying resolutions. The relationship between obtained static contact angle and the value of s_{sf} is largely linear between angles $180° - 55°$ and kink in the graph occurs around angle $55°$ and is linear again further on. From the above, Figure 6.5, it is clear that the particle resolution does not effect much the relation between the static contact angle, θ, and the interaction force between the solid and fluid, s_{sf}. This can be seen in Figure 6.6 where the surface profiles for the three resolutions of 528, 900 and 1940, at s_{sf} of 0.0005, 0.0035 and 0.008, respectively, were compared. It can be seen that the free-surface profiles are almost overlapping.

The droplet shape at the stationary state becomes more symmetric for the smaller static angles by increasing the droplet resolution as shown in Figure 6.6c.

In the proposed IIF method, the Tait's equation of state is limited to static contact angles between around 30° and 180°, while Kordilla *et al.* [79] achieved between 25° and 110° with Van der Waals equation of state. Now, the Tait's equation of state does not require any tuning parameter unlike the Van der Waals equation of state which has three parameters, that has to be tuned in order to mimic the correct physics and is not a straightforward approach. Furthermore, the proposed approach uses only a single phase model by ignoring the surrounding air while the original approach uses two phases which makes it more computationally expensive.

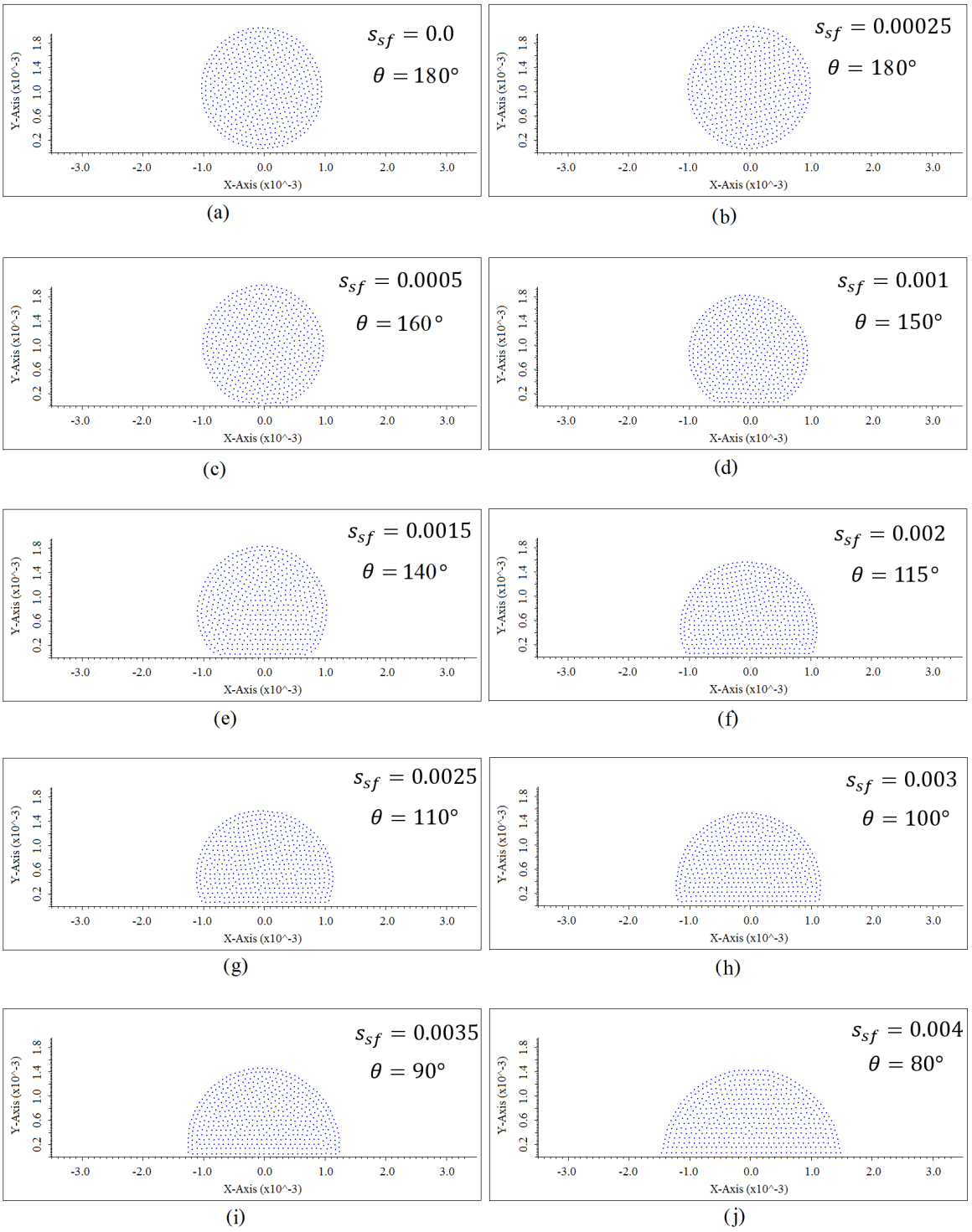
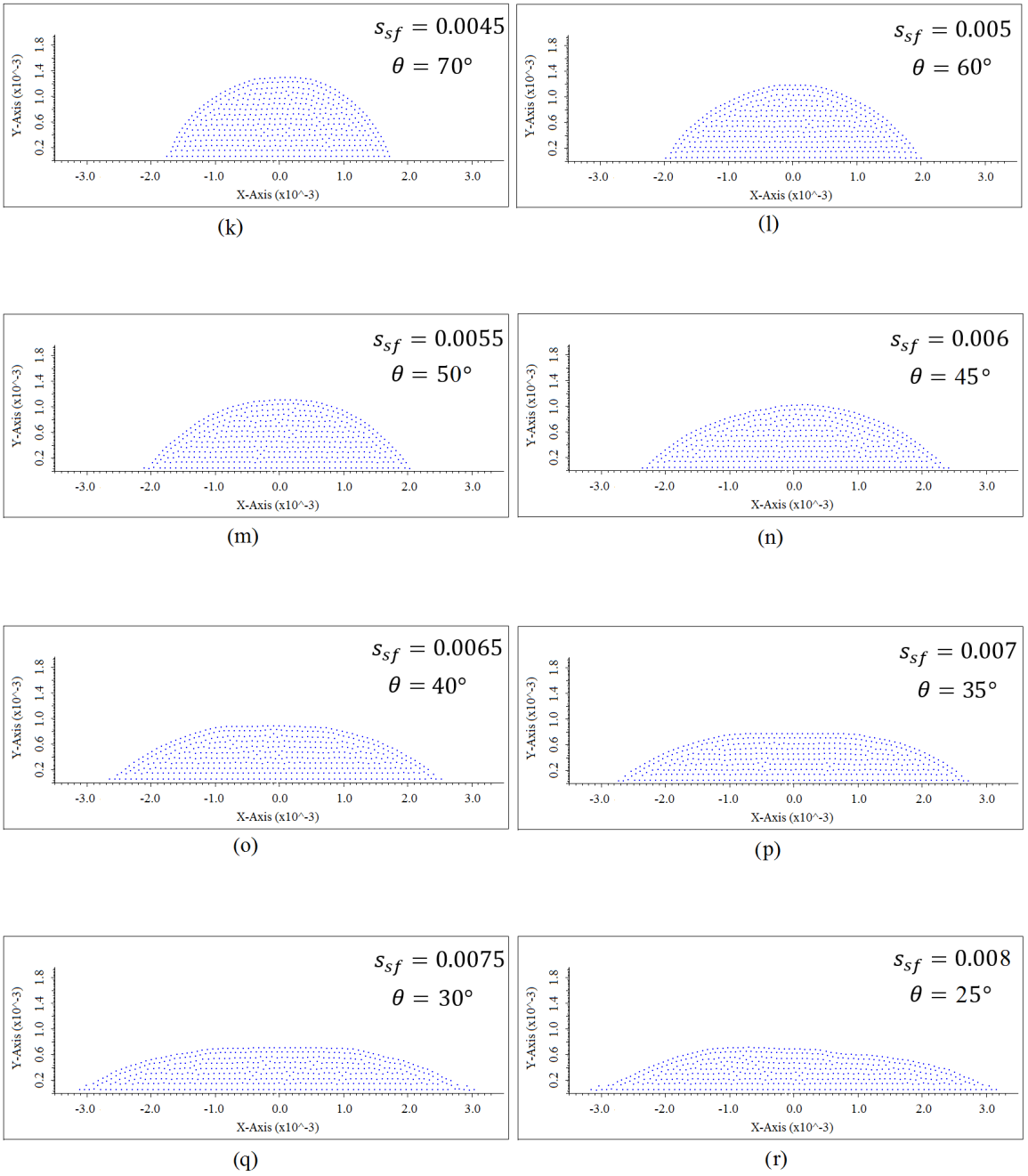


FIGURE 6.2: Contact angle dependance on the interaction force between the solid and fluid with the fluid particle resolution of 528. The solid substrate is not shown for simplicity reason.



Continuation of Figure 6.2: Contact angle dependance on the interaction force between the solid and fluid with the fluid particle resolution of 528. The solid substrate is not shown for simplicity reason.

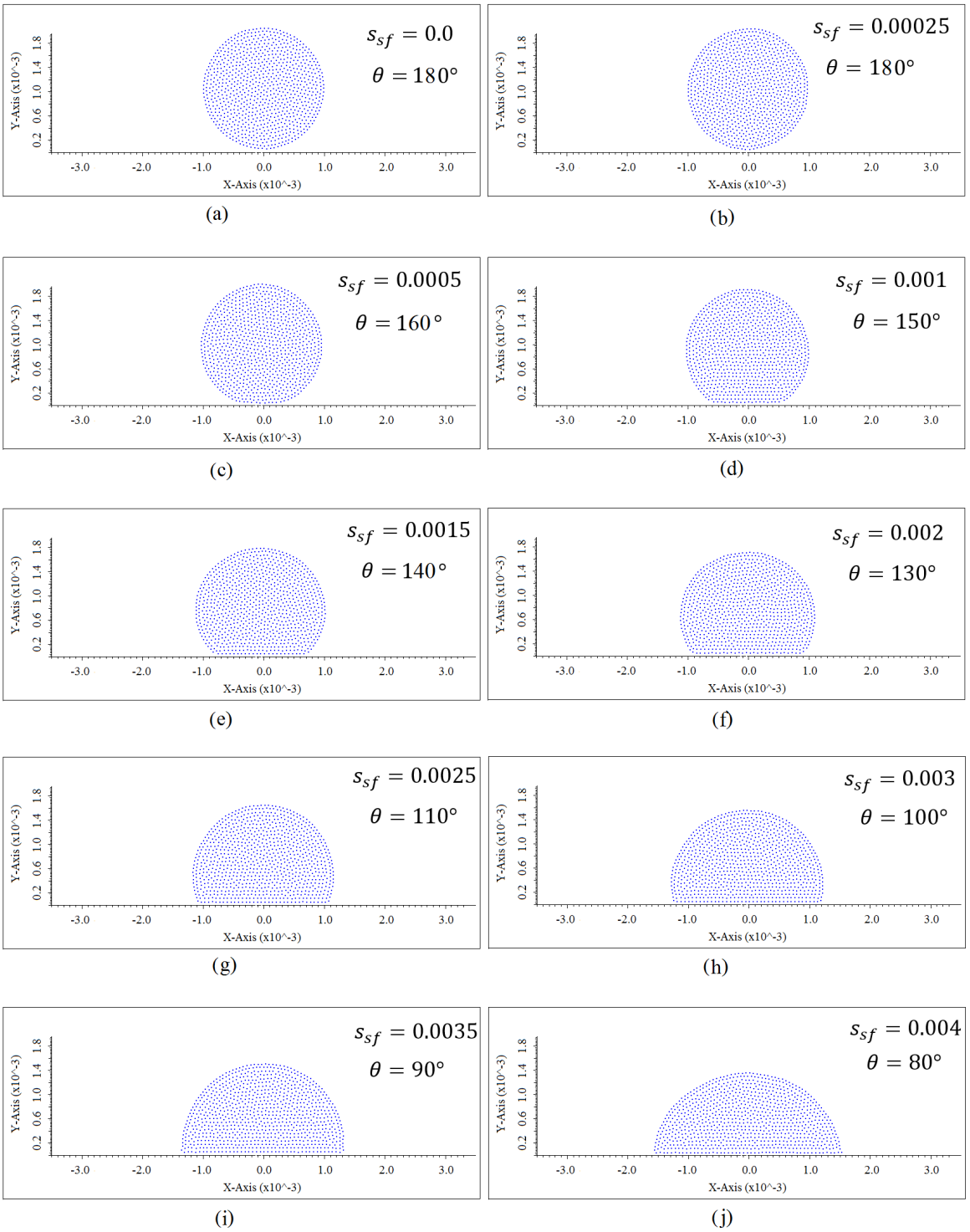
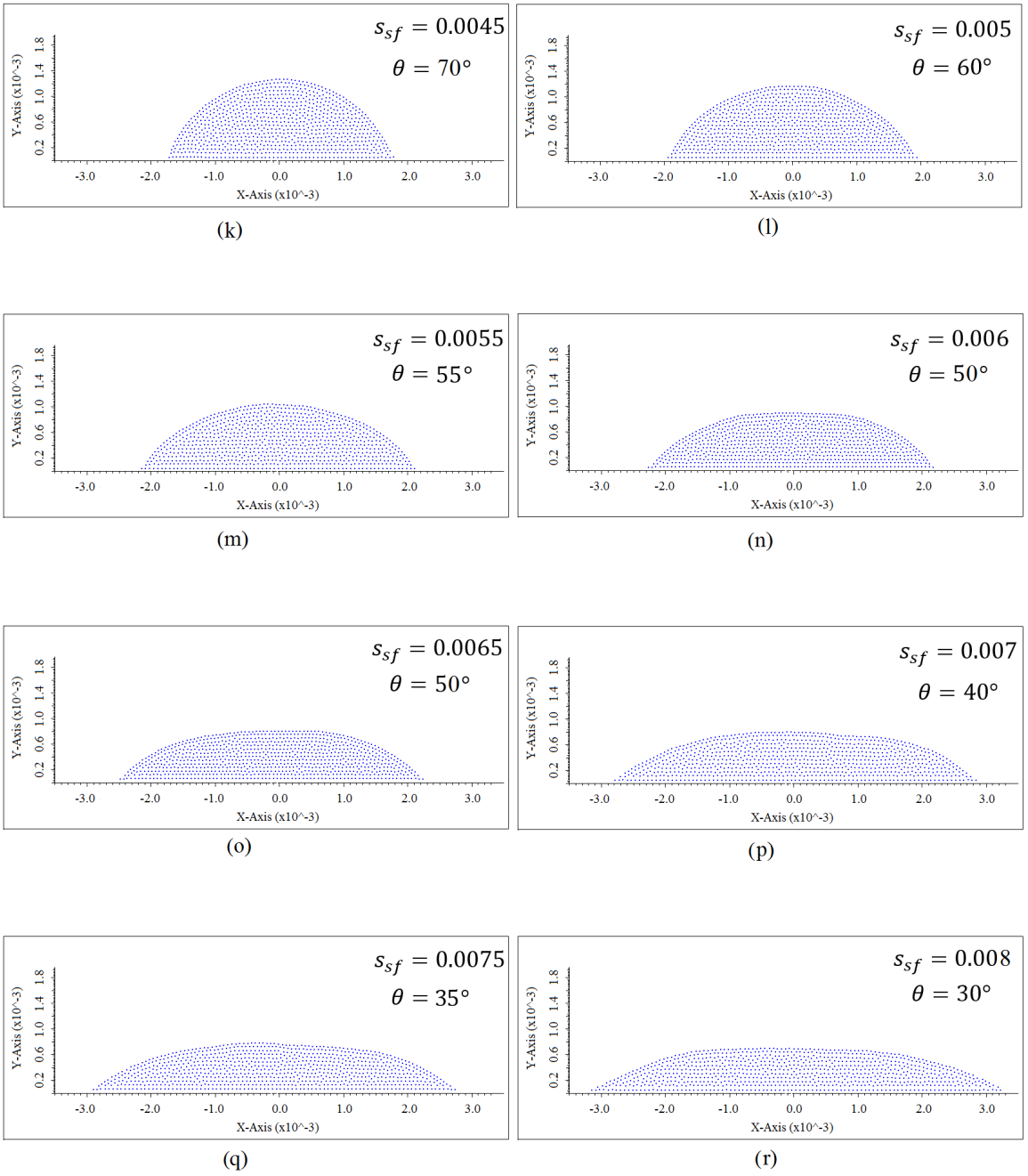


FIGURE 6.3: Contact angle dependance on the interaction force between the solid and fluid with the fluid particle resolution of 900. The solid substrate is not shown for simplicity reason.



Continuation of Figure 6.3: Contact angle dependance on the interaction force between the solid and fluid with the fluid particle resolution of 900. The solid substrate is not shown for simplicity reason.

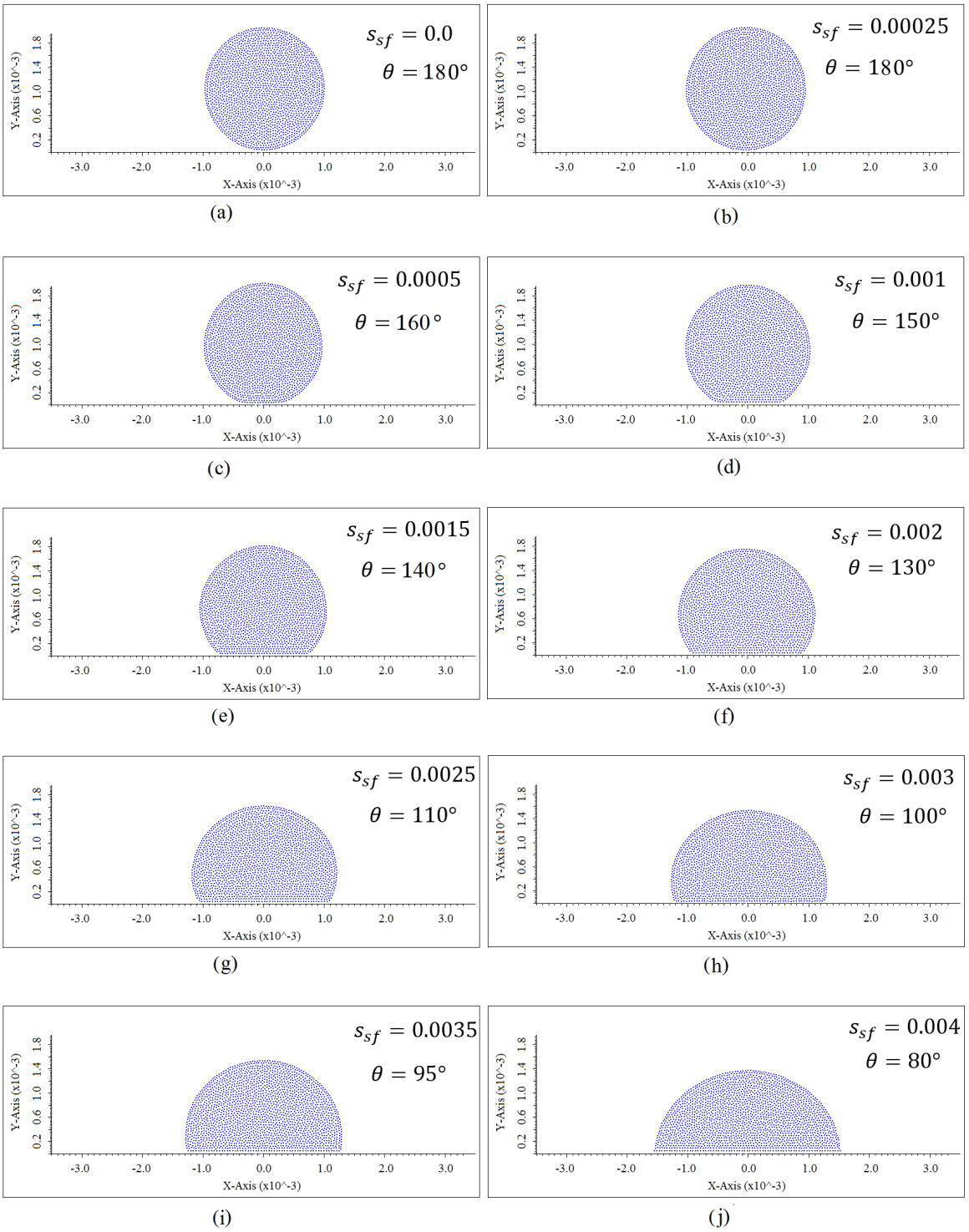
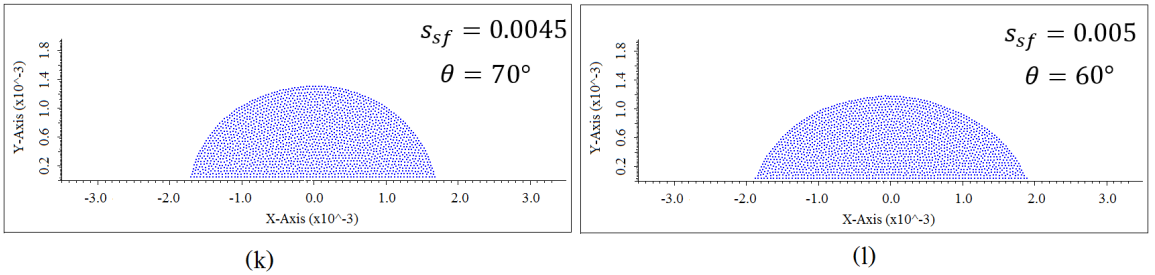
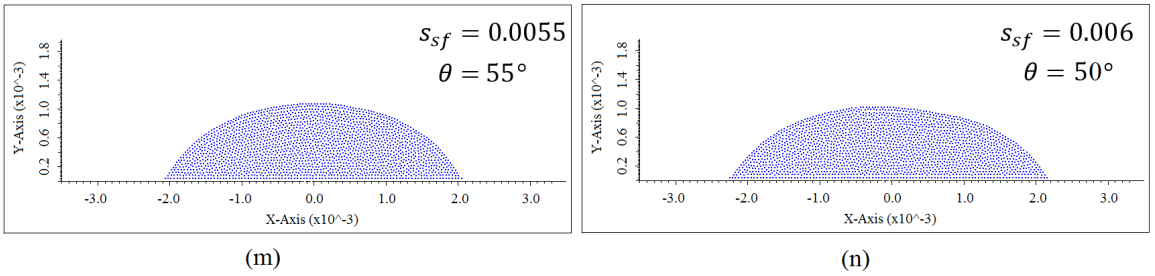


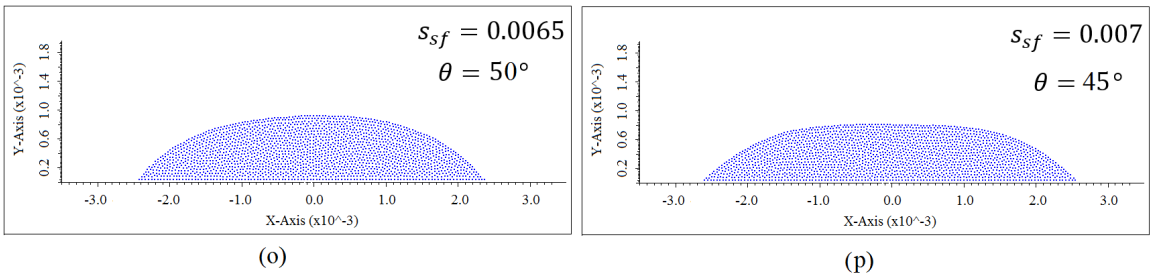
FIGURE 6.4: Contact angle dependance on the interaction force between the solid and fluid with the fluid particle resolution of 1940. The solid substrate is not shown for simplicity reason.



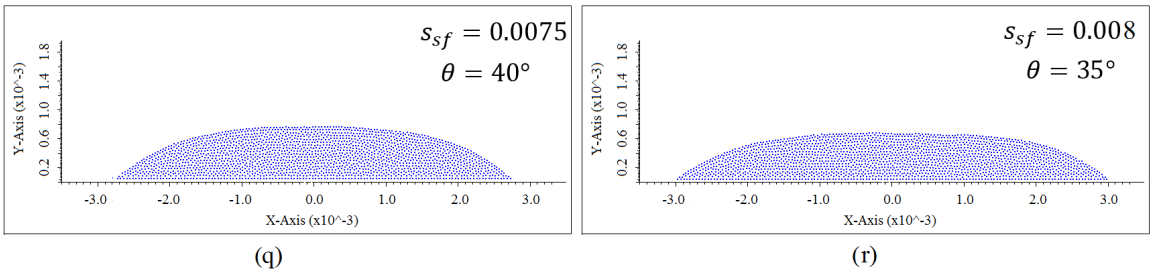
(k) (l)



(m) (n)



(o) (p)



(q) (r)

Continuation of Figure 6.4: Contact angle dependance on the interaction force between the solid and fluid with the fluid particle resolution of 1940. The solid substrate is not shown for simplicity reason.

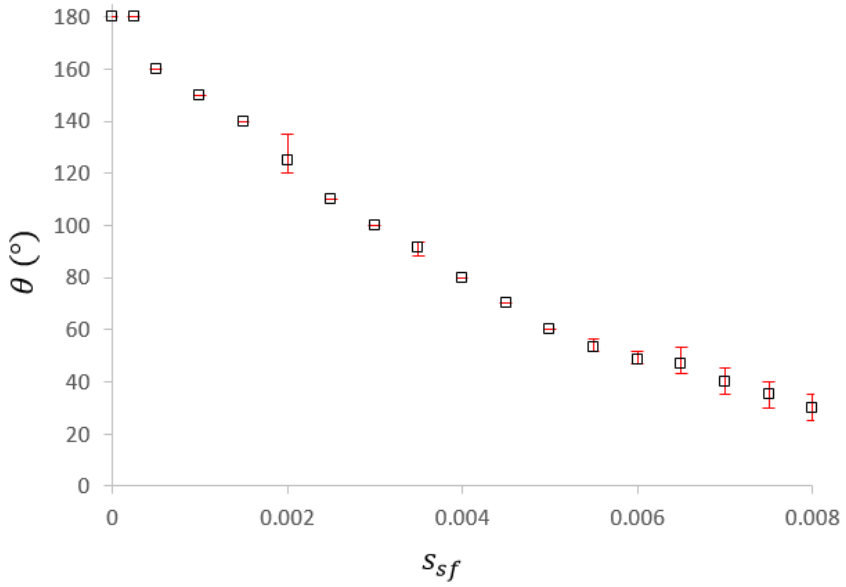


FIGURE 6.5: Mean static equilibrium contact angle (θ) for different interaction force between the fluid and solid (s_{sf}) and standard deviation.

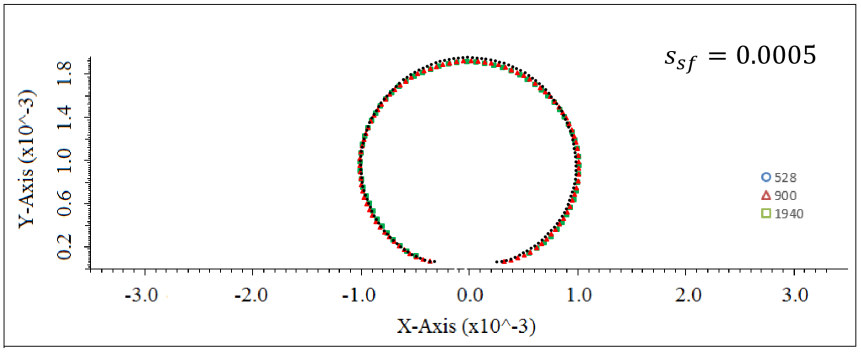
6.1.2 CLF method

Recently, Huber *et al.* [74] proposed the so called Contact Line Force (CLF) model to control the contact angle for two-phase problems. This force is only applied to the free-surface fluid particles near the solid substrate (see Figure 6.7a) while the CSF is applied to the other surface fluid particles far away from the solid substrate (see Figure 6.7b). This method is simple and easy to implement into the Navier-Stokes equation as an external force. The idea behind the CLF method is that this external force is applied parallel to the solid substrate as shown in Figure 6.7a.

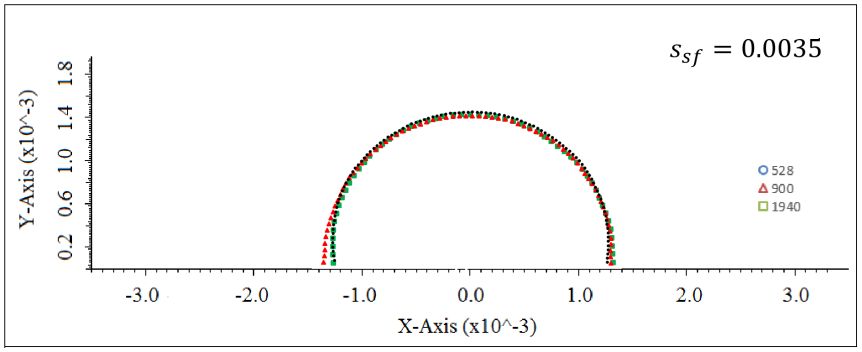
The CLF per unit mass for a two phase problem is given by:

$$\mathbf{f_c} = \frac{\sigma\Big[cos(\alpha_s) - cos(\alpha_d)\Big]\hat{\nu}}{\rho}\delta \tag{6.1}$$

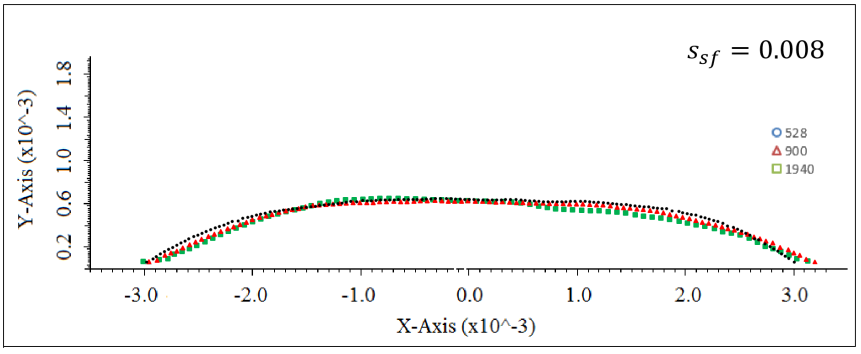
where α_s is the prescribed static contact angle, α_d is the dynamic contact angle, σ is the surface tension, ρ is the density and $\hat{\nu}$ is the unit vector with direction parallel to the substrate (see Figure 6.9). The present study proposed a modification to the



(a)



(b)



(c)

FIGURE 6.6: Comparison of droplet shapes at different particle resolutions for the strength of force between the solid and fluid particles s_{sf} at a) 0.0005, b) 0.0035, c) 0.008.

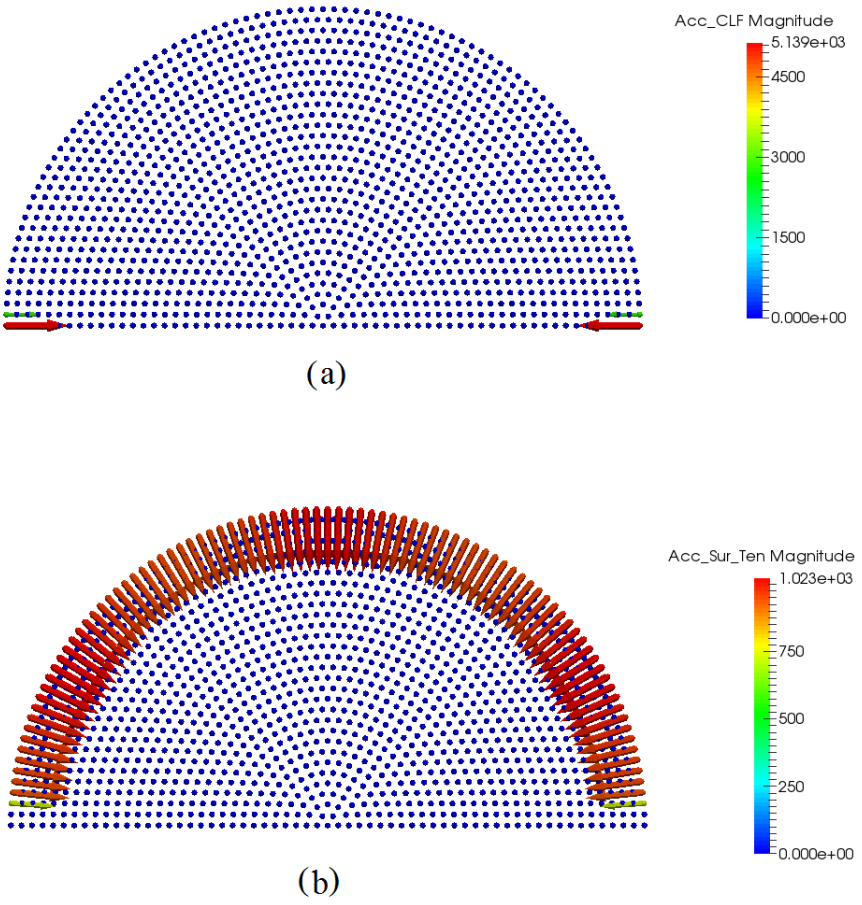


FIGURE 6.7: Initial particle distribution of the semicircle droplet on the substrate (for simplicity the substrate is not shown here): a) The CLF. b) The modified CSF.

above Equation (6.1) for a single phase model, thus giving the following:

$$\mathbf{f_c} = \gamma \frac{\sigma \Big[cos(\alpha_s) - cos(\alpha_d) \Big] \hat{\nu}}{\rho} \delta \tag{6.2}$$

where $\gamma = 2.0$ which will be verified in Subsection 6.1.3 and all other parameters remain the same.

At the equilibrium state, $\mathbf{f_c} = 0$, because the static and dynamic contact angles balances each other and, if otherwise, the interface at the droplet will evolve until $\mathbf{f_c} = 0$ is satisfied. In the current model, only the static contact angle is specified.

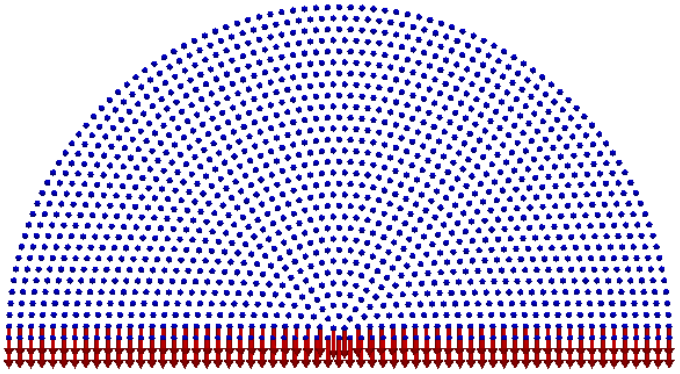


FIGURE 6.8: The direction of the unit distance vector for the fluid particles when the summation is only applied over the solid boundary particles.

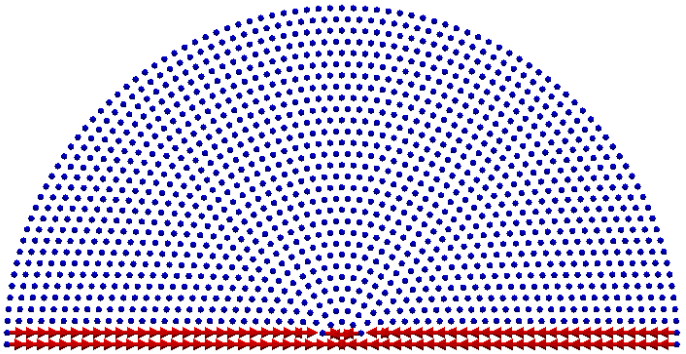


FIGURE 6.9: The direction of the unit vector for the fluid particles near the solid boundary.

The dynamic contact angle is the actual (real) contact angle and is computed from:

$$cos(\alpha_d) = -\hat{\mathbf{d}}_i \cdot \hat{\mathbf{n}}_i \tag{6.3}$$

where $\hat{\mathbf{d}}_i$ is the unit distance vector and $\hat{\mathbf{n}}_i$ is the unit surface normal (see Equation (5.6)).

The distance vector is given by:

$$\mathbf{d_i} = \sum_j V_j \mathbf{r_{ij}} W_{ij} \tag{6.4}$$

The distance vector points direct to the substrate where the summation is only introduced over the solid particles as shown in Figure 6.8. The vector in the right hand

side of the Equation (6.1) is calculated from the following equation:

$$\nu_{\mathbf{i}} = |\mathbf{d_i}|^2 \mathbf{n_i} - (\mathbf{d_i} \cdot \mathbf{n_i})\mathbf{d_i} \tag{6.5}$$

Figure 6.9 shows direction of the unit vector for the fluid particles near the solid substrate since the distance vector $\mathbf{d_i}$ exists only for these fluid particles. Further, these unit vectors are used as direction of the CLF for the edge fluid particles at the interface at both side.

The aim of δ in the Equation (6.1) is to transform the the force per line to the force per volume. This function is given by:

$$\delta_i = -2\hat{\mathbf{d}}_i \cdot \sum_j V_j(\delta'_j - \delta'_i)\nabla W_{ij} \tag{6.6}$$

with

$$\delta'_j = \begin{cases} \delta'_i & \text{if } j \in \text{ fluid} \\ 0 & \text{if } j \in \text{ boundary} \end{cases} \tag{6.7}$$

and

$$\delta'_i = \hat{\nu}_{\mathbf{i}} \cdot \mathbf{n_i} \tag{6.8}$$

As it can be seen from the Equation (6.1), only two parameters, the surface tension coefficient, σ, and the static contact angle, α_s, are used as an input parameters to control the contact angle.

6.1.3 Droplet Spreading using CLF

For this section, the initial shape of the droplet is started from semicircle since as stated earlier in Subsection 5.2.2) square shape has edge issues which will further compounds the spreading process. For the setup, an initial semicircle of 1.5 mm with 1420 fluid particles is placed on a flat substrate with 490 solid particles. The substrate is modelled using five layers of dummy particles with the same initial parameters as with the fluid particles. A Wendland kernel is used with smoothing length, $h = 1.3dx$,

where dx is a initial separation of the particles, which is constant during the simulation for all particles. The reference sound speed was chosen to 50 times maximum velocity during the simulation which is $\sqrt{2gR}$ where R is the radius of the droplet and g is the gravity. The Verlet scheme is used to update the position and the velocity of the SPH particles. Continuity density approach is used to update the density field with a reference density $\rho_0 = 1000kg/m^3$. Pressure is computed with the Taits equation of state. All other initial properties are shown in Table 6.2. The surface tension effect was simulated with the modified CSF method while the contact angle was carried out with the CLF method.

The challenging part of CLF method is the computation of the dynamic contact angle. The reason why it is challenging is that the unit surface normal in Equation (6.3) points in the wrong direction for the surface fluid particle near the solid substrate since it involves the summation of all neighbouring particles (see Equation (5.6)) which in this case lead to the wrong dynamic contact angle computation as shown in Figure 6.10a. Based on the initial starting shape of the droplet on the flat substrate (see Figure 6.10), the dynamic contact angle is 90° (see Figure 6.10b with green shaded semicircle) which can be achieved by considering the correct neighbour region. To get the correct neighbour list for the surface fluid particles near the substrate, simple algorithm were developed and applied. Conditions $\nabla \cdot \mathbf{r} < 1.5$ (see Equation (5.5)) and $|\mathbf{d_i}| \neq 0$ (see Equation (6.4)) are used to track the free-surface fluid particles close the solid substrate in order to apply the CLF conditions. The applied CLF will either drive the contact line away (see Figure 6.11a) or towards (see Figure 6.11b) the bulk of the fluid when the static contact angle is smaller or bigger than the initial dynamic contact angle of 90°. In either case, the contact line will begin to move continuously until the dynamic contact angle equals to the prescribed static equilibrium contact angle, so that the condition $\mathbf{f}_c = 0$.

Figures 6.12, 6.13 and 6.14 show the droplet evolution shapes at equilibrium state for various static contact angles with the fluid particle resolutions of 375, 1000 and 3880, respectively. It can be seen from these Figures that the droplet shapes are realistic and symmetric on both sides. However, for some contact angles the maximum

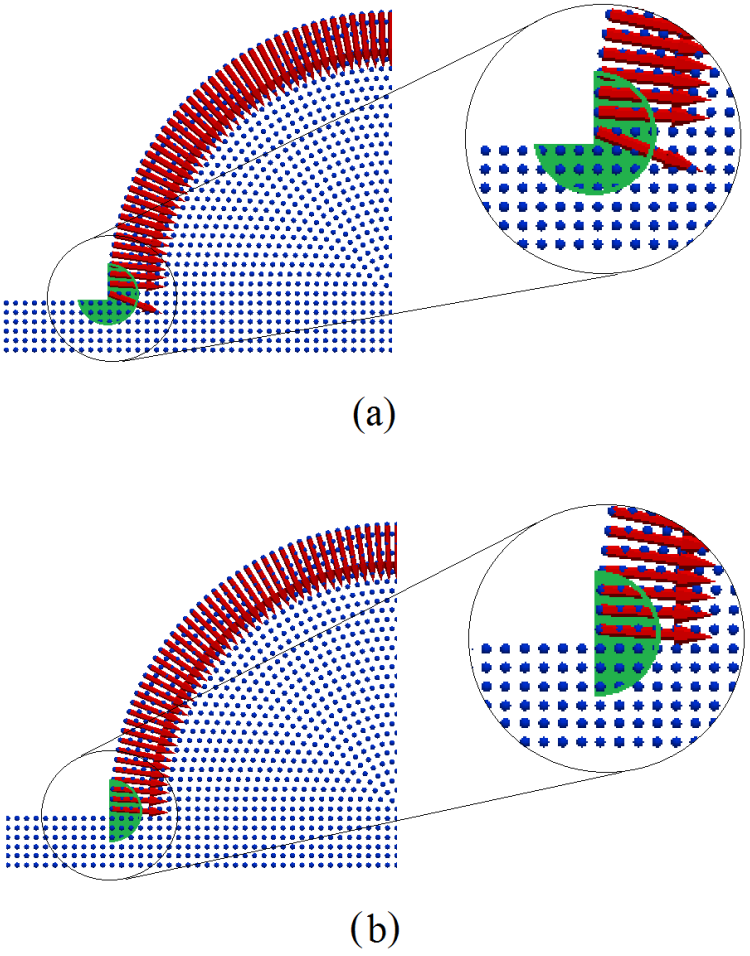
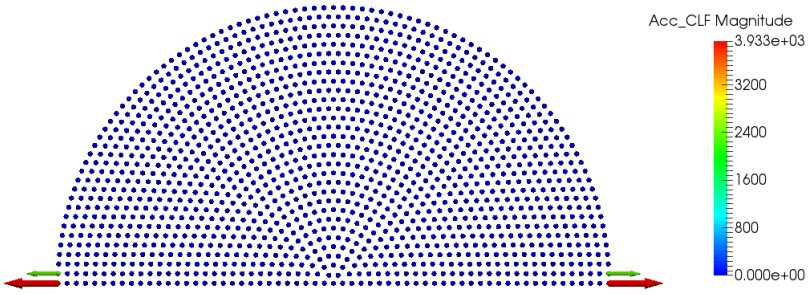


FIGURE 6.10: The direction of the unit surface normal for the surface fluid particles near the solid substrate: a) with the default calculation. b) expected

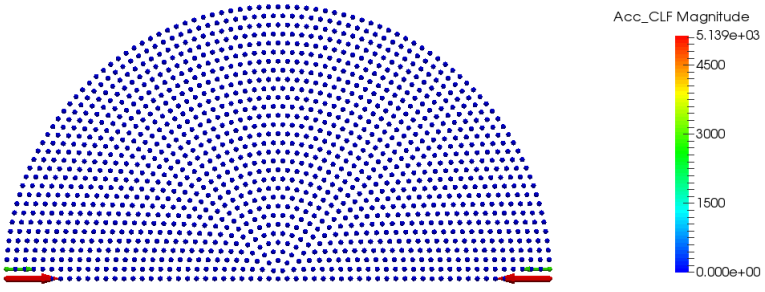
absolute error is around 10% as shown in Figure 6.16. Huber *et al.* [74], in their work had an absolute error of around 5% by applying the method to two-phase problems. Absolute error decreases when the fluid particle resolution increases for the contact angles between 30° and 110° as shown in Figures 6.16 and 6.15b-c. It can be seen that the so called "mushroom" formation is more obvious as a result of particle deficiency noticed at lower particle resolutions as shown in Figures 6.12a-c and 6.13a-c. From the present work where the CLF method is applied for single-phase problem for the first time, the difference between the static contact angle and the set static contact angle is considered to be minimal.

	Fluid
Density, ρ (kg/m^3)	1000
Pressure, p (Pa)	0
Velocity, $\vec{v}$ (m/s)	0
Mass, m (kg)	$1.4 \cdot 10^{-6}$
Acceleration, $\dot{\vec{v}}$ (m/s^2)	0
Initial particle separation, $dx = dy$ (m)	$3.75 \cdot 10^{-5}$
Smoothing length, h (m)	$1.3dx$
Surface tension coefficient, σ (N/m)	0.0182
Dynamic viscosity, μ $(Pa \cdot s)$	10^{-2}

TABLE 6.2: Initial parameters of the droplet.



(a)



(b)

FIGURE 6.11: The direction of the CLF depending on the static contact angle and the initial dynamic contact angle a) when the static contact angle is smaller than the initial dynamic contact angle of 90° b) when the static contact angle is higher than the initial dynamic contact angle of 90°

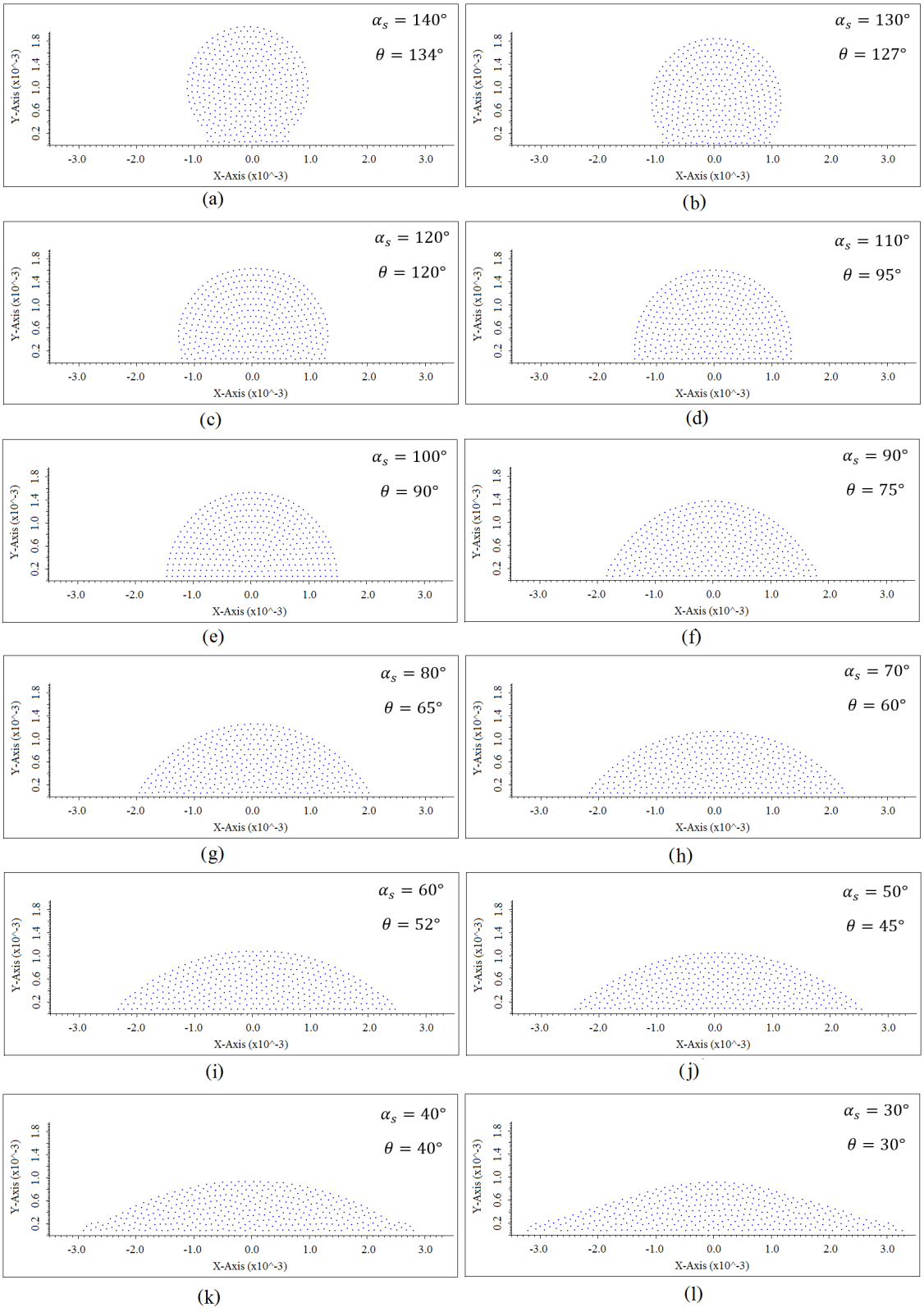


FIGURE 6.12: Droplet shapes at equilibrium state after applying the CLF with 375 fluid particles for various static contact angles. α_s and θ correspond to set contact angle and contact angle from simulation.

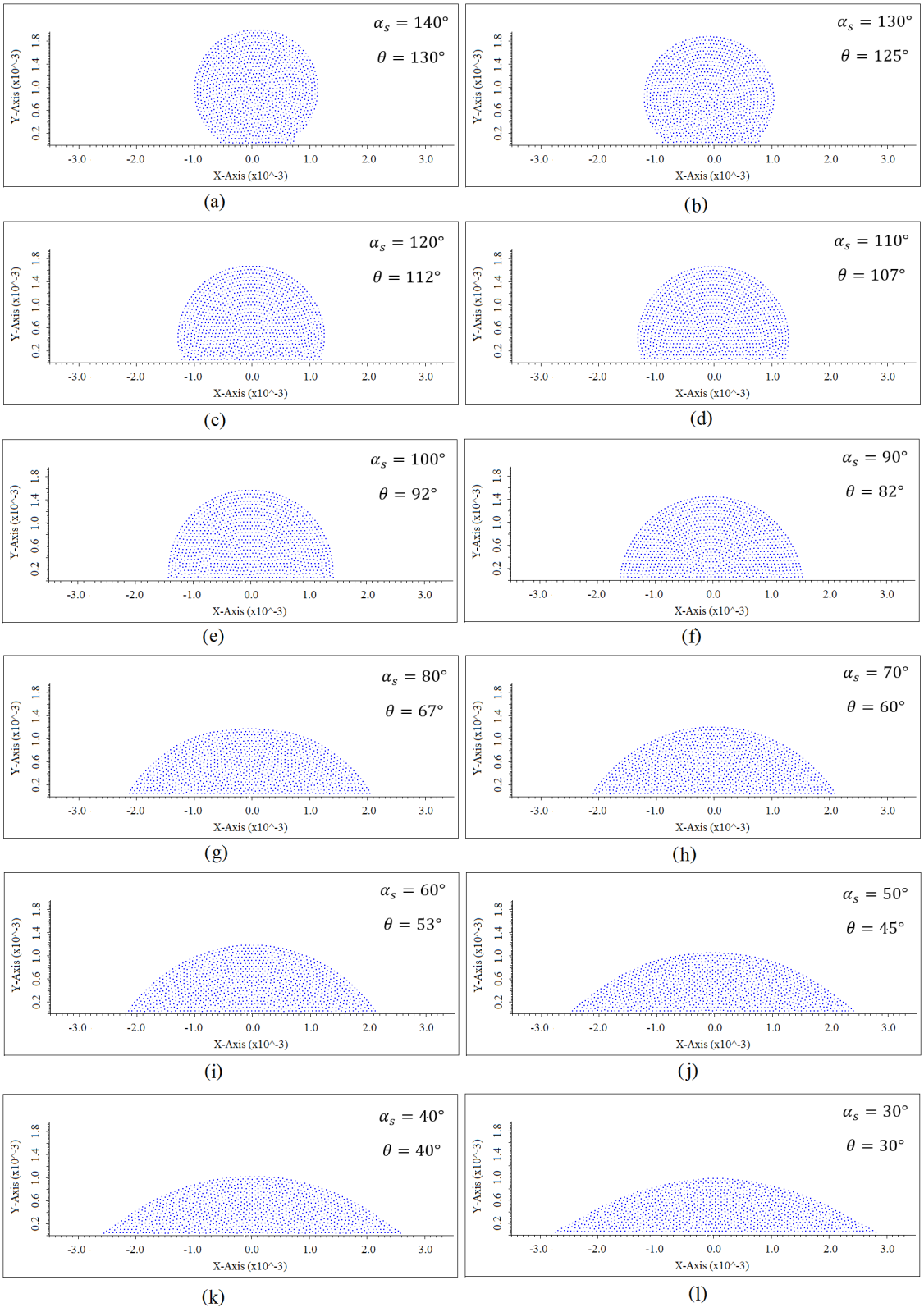


FIGURE 6.13: Droplet shapes at equilibrium state after applying the CLF with 1000 fluid particles for various static contact angles. α_s and θ correspond to set contact angle and contact angle from simulation.

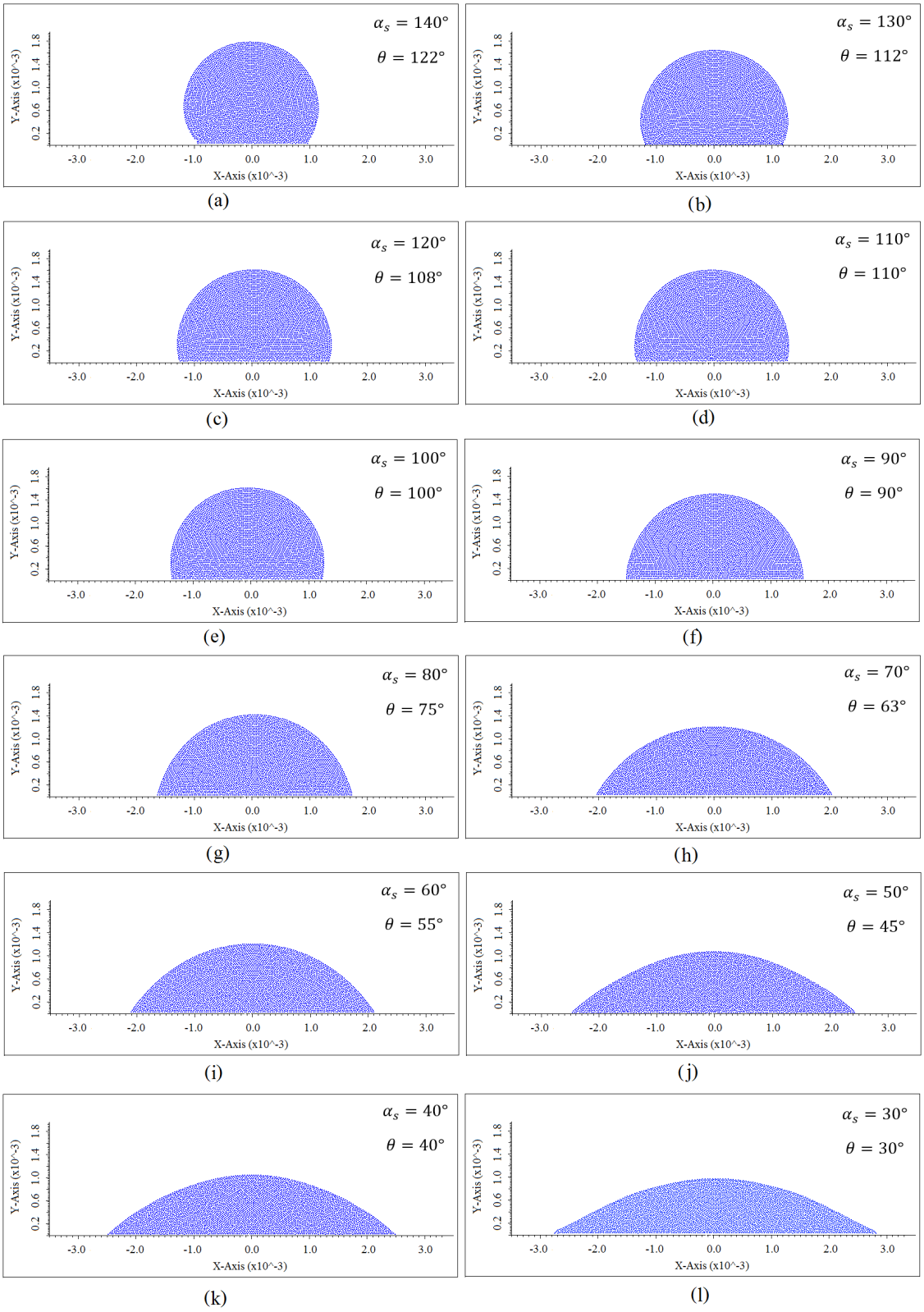
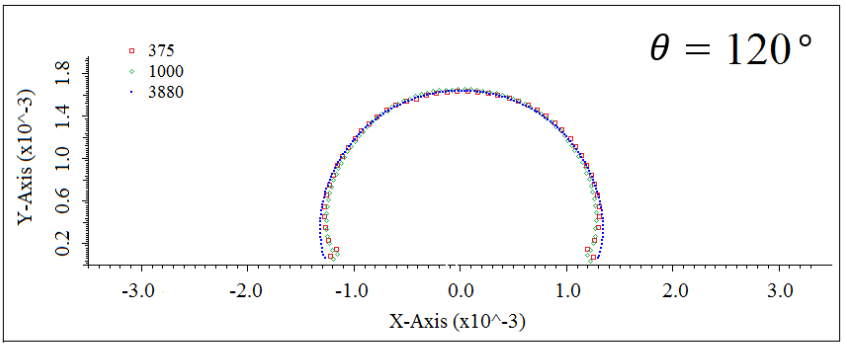
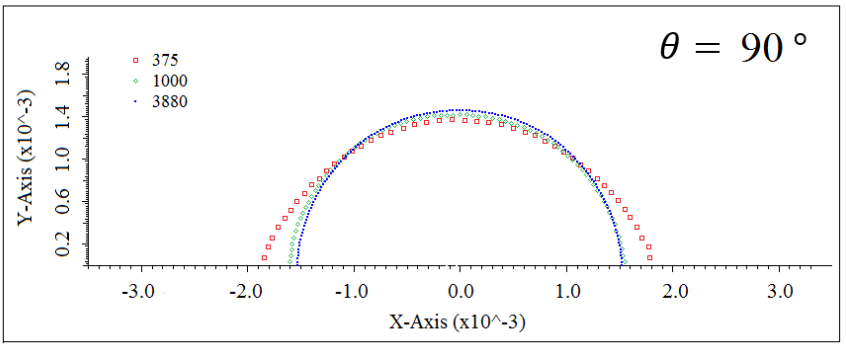


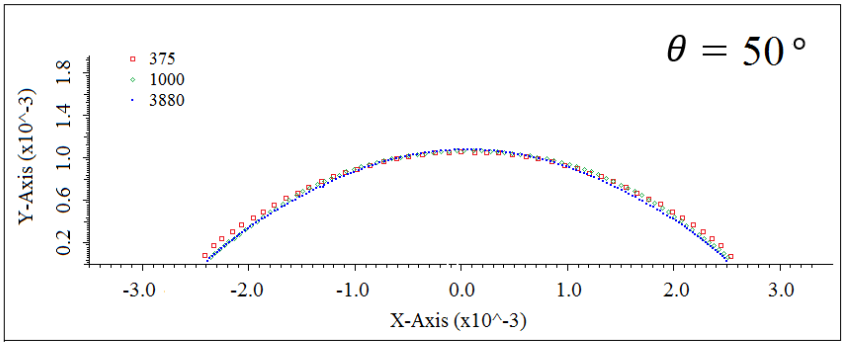
FIGURE 6.14: Droplet shapes at equilibrium state after applying the CLF with 3880 fluid particles for various static contact angles. α_s and θ correspond to set contact angle and contact angle from simulation.



(a)



(b)



(c)

FIGURE 6.15: Comparison of droplet shapes at different particle resolutions for the contact angles of a) 120°, b) 90°, c) 50°.

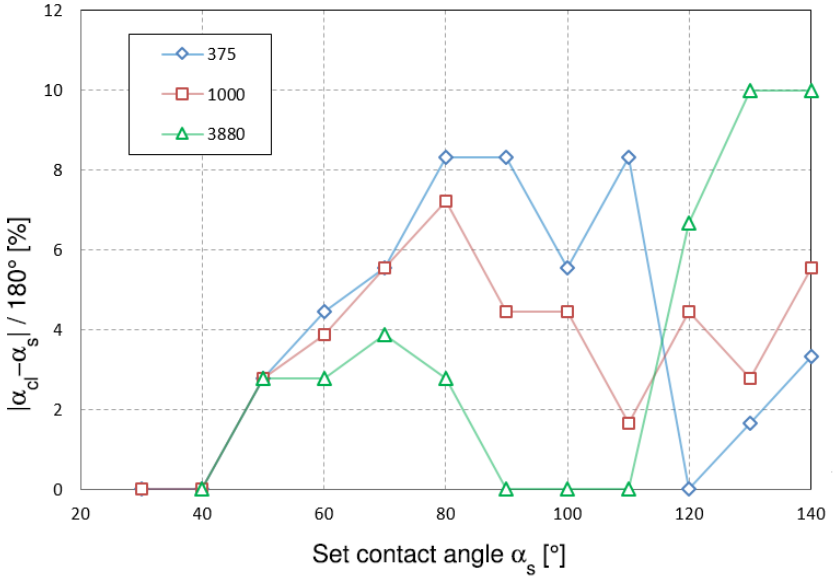


FIGURE 6.16: Absolute error of the static contact angle against the set contact angle with different particle resolutions

6.1.4 The Disjoining Pressure Method

In this subsection the proposed the so called "disjoining pressure" method to control the contact angle with SPH method is considered for the first time. Sellier [136] and Schwartz [135] used the disjoining pressure method to simulate the interface of thin film flow and droplets over homogenous and heterogeneous substrate using a mesh based lubrication approach. The disjoining pressure method is based on the hypothesis that a thin film exist at the contact line. This thin film is known as a precursor film and has been observed experimentally when the fluid completely wets the substrate. More information on the existence of the precursor film can be found in de Gennes [41].

Following Sellier [136], the disjoining pressure that is used in the present simulation is given by

$$\Pi(H) = B\left[\left(\frac{H^*}{H}\right)^n - \left(\frac{H^*}{H}\right)^m\right], \tag{6.9}$$

where B, n and m are constants and H^* the thickness of the precursor film which is believed to be between 1 to 100nm. The first term on the right hand side in Equation (6.9) describes the fluid-solid attraction and the second term describes the fluid-solid

repulsion. Different values for (n, m) have been used in the literatures. The values of $(3, 2)$ was used by Teletzke [148] and Churaev and Sobolev [34] while Mitlin and Petviashvili [105] used the values of $(9, 3)$. Figure 6.17 shows the normalised disjoining pressure with the constants $(9, 3)$ and $(3, 2)$ for (n, m).

According to Schwartz's [135], the constant B can be derived from a balanced equilibrium force at the contact angle area when the droplet reaches its stationary state. The value of B is assumed to be constant during the movement of the contact line. The detailed derivation for B is presented in Appendix B, and final form is given by:

$$B = \frac{(n-1)(m-1)}{H^*(n-m)}\sigma(1-\cos\theta) \tag{6.10}$$

where θ is equilibrium static contact angle.

The spreading rate relies on the choice of H^* which is arbitrary and defined as βdx. For this reason, the static contact angle of $\theta = 90°$ is set and β is tuned until an appropriate value for H^* achieved to produce the desired static equilibrium contact angle. Once the thickness of the precursor film is obtained, it remains constant for computing all the other contact angles. In the SPH method, there is no need to

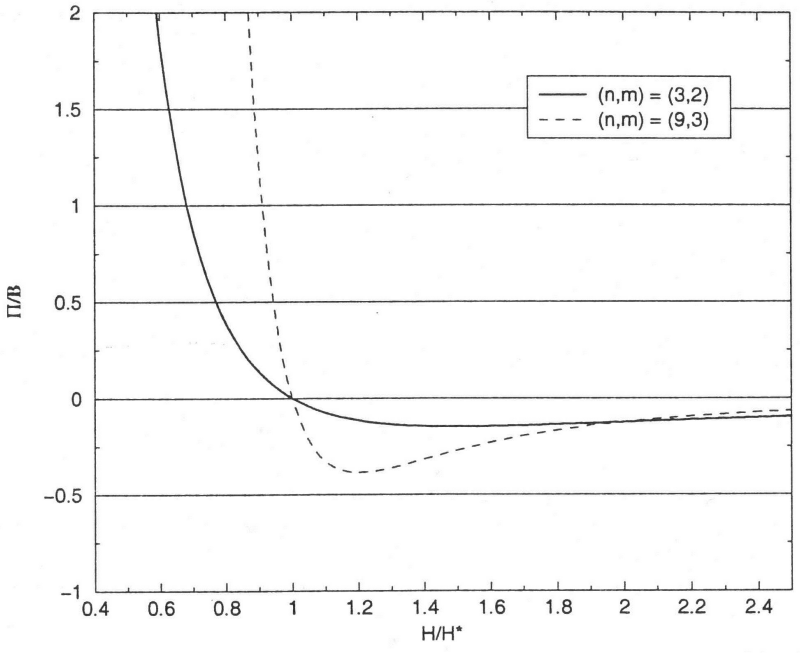


FIGURE 6.17: Normalised disjoining pressure for $(n, m) = (9, 3)$ and $(n, m) = (3, 2)$

generate a precursor film separately, this is because the substrate can be assumed as the precursor film since it carries the same fluid properties. Figure 6.18 shows the schematic representation of the pressure field on the droplet and the substrate (precursor film). The disjoining pressure is applied to the precursor film which begins from the interface contact line in both directions (see Figure 6.18 highlighted in blue). Tait's equation is applied to the all the fluid particles while the disjoining pressure is applied only to the free-surface particles on the fluid (see Figure 6.18 highlighted in red). Furthermore, the substrate is generated in staggered manner in order to achieve more uniform movement of the contact line.

For the setup, an initial semicircle of 1.5 mm with 425 fluid particles is placed on 308 solid substrate particles. A Wendland kernel is used with smoothing length of $h = 1.3dx$ which is constant during the simulation for all particles, where dx is a initial separation of the particles. The sound speed was chosen to 50 times maximum velocity calculated from $\sqrt{2gR}$ where R is the radius of the droplet and g is the gravity. The Verlet scheme is used to update the position and the velocity of the SPH particles. Continuity density approach is used to update the density field with a reference density, $\rho_0 = 1000$ kg/m^3. Pressure is computed with the Taits equation of state. All other initial properties are shown in Table 6.3.

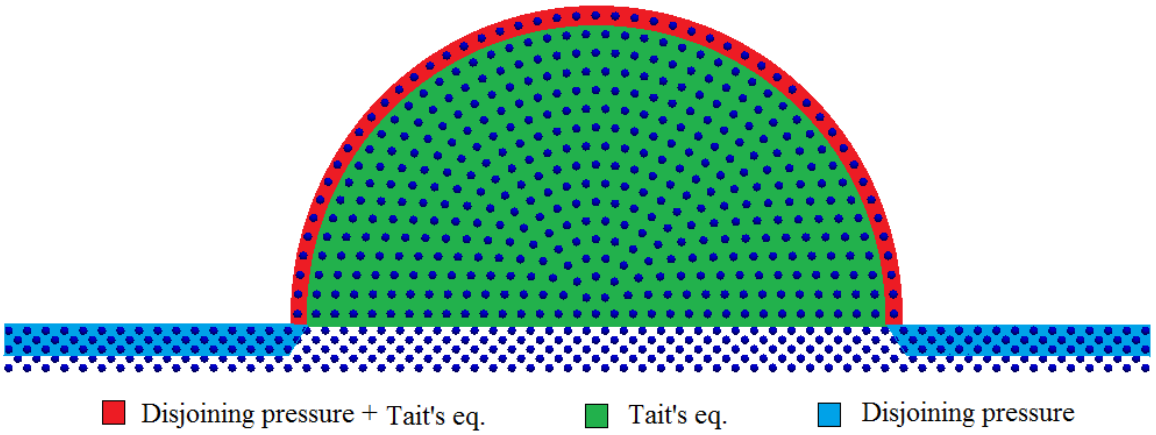


FIGURE 6.18: The pressure profile for the droplet and the precursor film. The disjoining pressure is applied to the precursor film and also to the surface of the droplet along with the Tait's equation.

	Fluid
Density, ρ (kg/m^3)	1000
Pressure, p (Pa)	0
Velocity, $\vec{v}$ (m/s)	0
Mass, m (kg)	$1.4 \cdot 10^{-6}$
Acceleration, $\dot{\vec{v}}$ (m/s^2)	0
Initial particle separation, $dx = dy$ (m)	$3.75 \cdot 10^{-5}$
Smoothing length, h (m)	$1.3dx$
Surface tension coefficient, σ (N/m)	0.072
Dynamic viscosity, μ $(Pa \cdot s)$	10^{-3}

TABLE 6.3: Initial parameters of the droplet.

Figure 6.19 shows the equilibrium static contact angles with $H^* = 2.8 \cdot dx$ for the droplet resolution of 425. A small gap between the droplet and the substrate can be seen for the contact angles that bigger than 90° (see Figure 6.19a-c). However, this gap decreases in size when the droplet resolution increases as shown in Figures 6.20a-c and 6.21a-c for resolution of 1000 and 1920, respectively, while keeping the same droplet size. The value of H^* is tuned to $3.1dx$ and $3.5dx$ for particle resolutions of 1000 and 1920, respectively, in order to achieve the required contact angle using the same contact angle as an input parameter, θ. The contact angle becomes smoother by increasing the particle resolution as it can be seen in surface profile comparison in Figure 6.22.

According to the Figure 6.24, it can be conclude that the H^* is directly proportional to the particle resolution. This proposed disjoining pressure method is limited to static contact angles between around 30° and 130°. The most important advantage of this method is that there no need to track the contact line like in the CLF method and therefore requires little computational time and resources.

6.1.5 Tanner's Law

Three different models (IIF, CLF and disjoining pressure) were proposed and employed to accurately predict the advancing/receding contact line at the interface of a droplet spreading on a homogeneously smooth flat substrate. In each model the accuracy of the approach is validated using Tanner's law [144] given by $H \sim t^{-1/\zeta}$ via measuring the change in droplet height, H, as a function of time, t; where the exponent for a two-dimensional droplet $\zeta = 7$. Also, an experiment that were verified by Lelah and Marmur's [82] found that the drop height as a function of time is given by $H \sim t^{-1/\zeta}$ where the constant parameter was found to varies between $0.16 \leq 1/\zeta \leq 0.32$.

Presently, for each model, an initial droplet with radius of 1.5 mm and resolution 1000 fluid particles is simulated by placing on a homogeneous flat substrate of a partial wetted fluid and fluid used is water. Initially, the droplet is allowed to stabilised at the set static contact angle of 60° after which it is allowed to spread by changing the set static contact angle to 30°. Figure 6.23 shows good agreement between decreasing H as a function of time for the different models employed and Tanner's law. It shows that the CLF and disjoining pressure model gives $1/\zeta$ a value for 0.138 and 0.137, respectively, which is almost exactly of that provided theoretically by Tanner's law of $1/\zeta = 0.14$. However, for the IIF model the value of $1/\zeta$ obtained is 0.359 which is still acceptable considering works presented by Lelah and Marmur's [82].

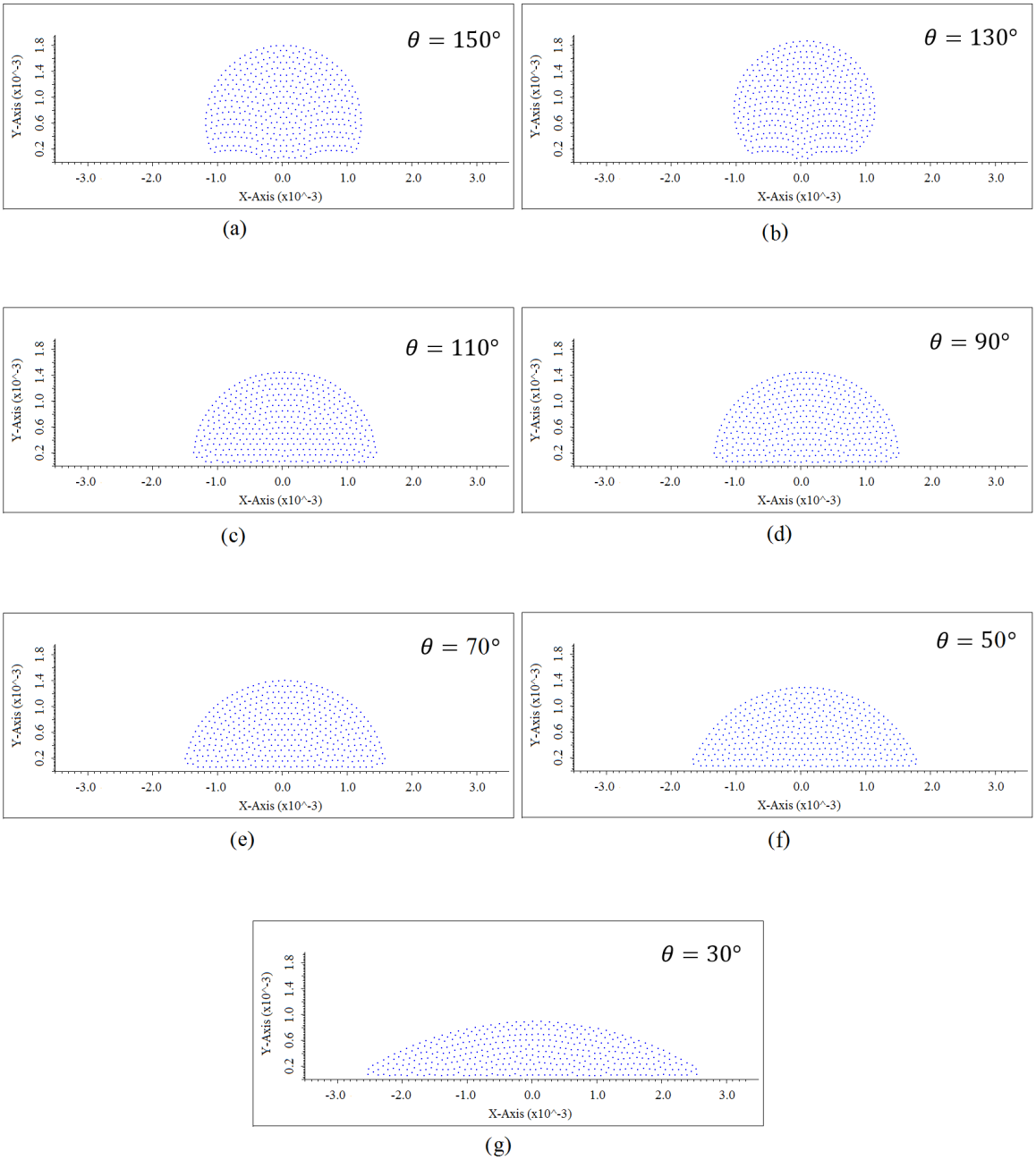


FIGURE 6.19: Contact angle dependance on the set contact angles with the fluid particle resolution of 424 and with $H^* = 2.8dx$. Static contact angles set to a) 150°, b) 130°, c) 110°, d) 90°, e) 70°, f) 50° and g) 30°

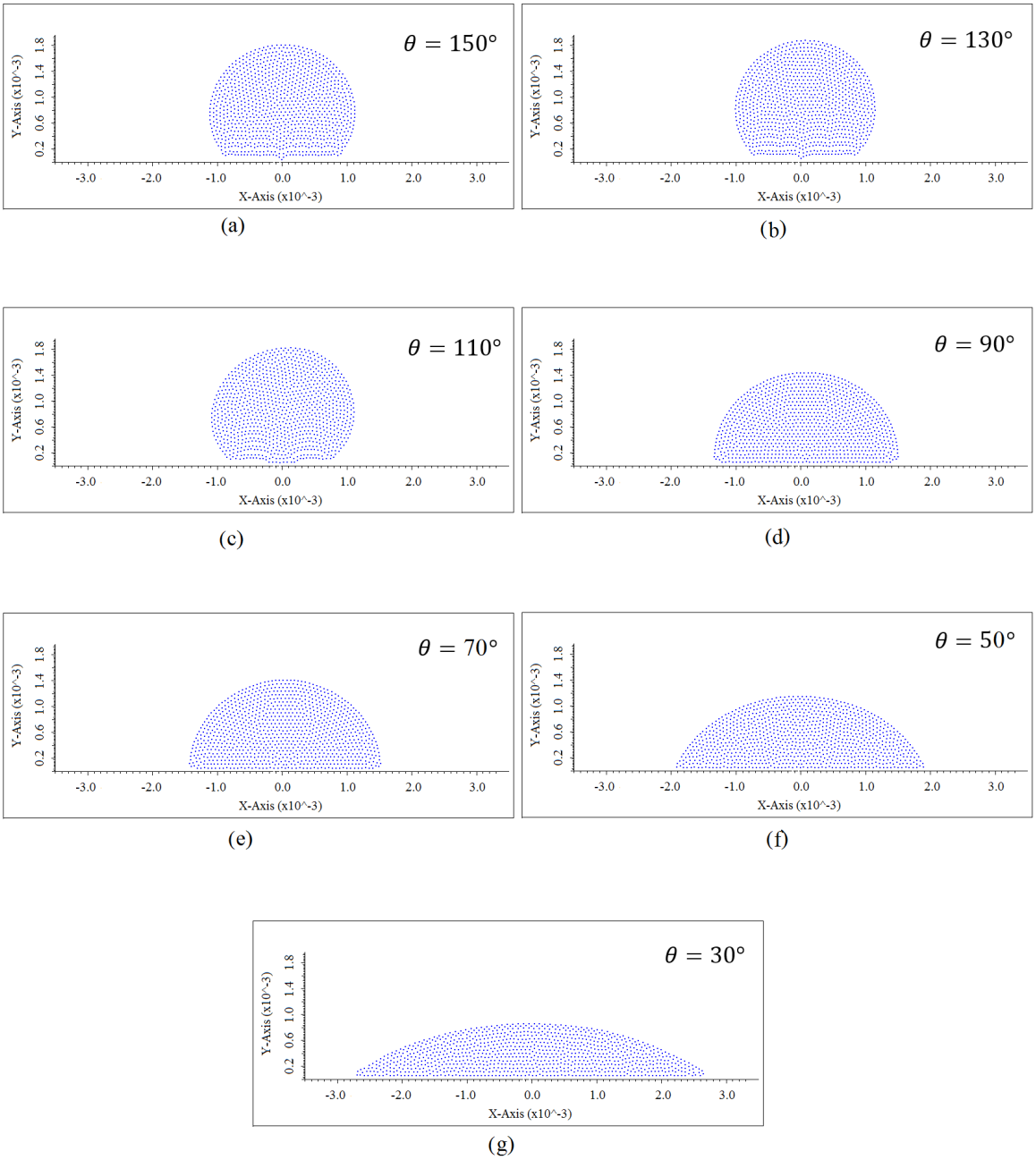


FIGURE 6.20: Contact angle dependance on the set contact angles with the fluid particle resolution of 1000 and with $H^* = 3.1dx$. Static contact angles set to a) 150°, b) 130°, c) 110°, d) 90°, e) 70°, f) 50° and g) 30°

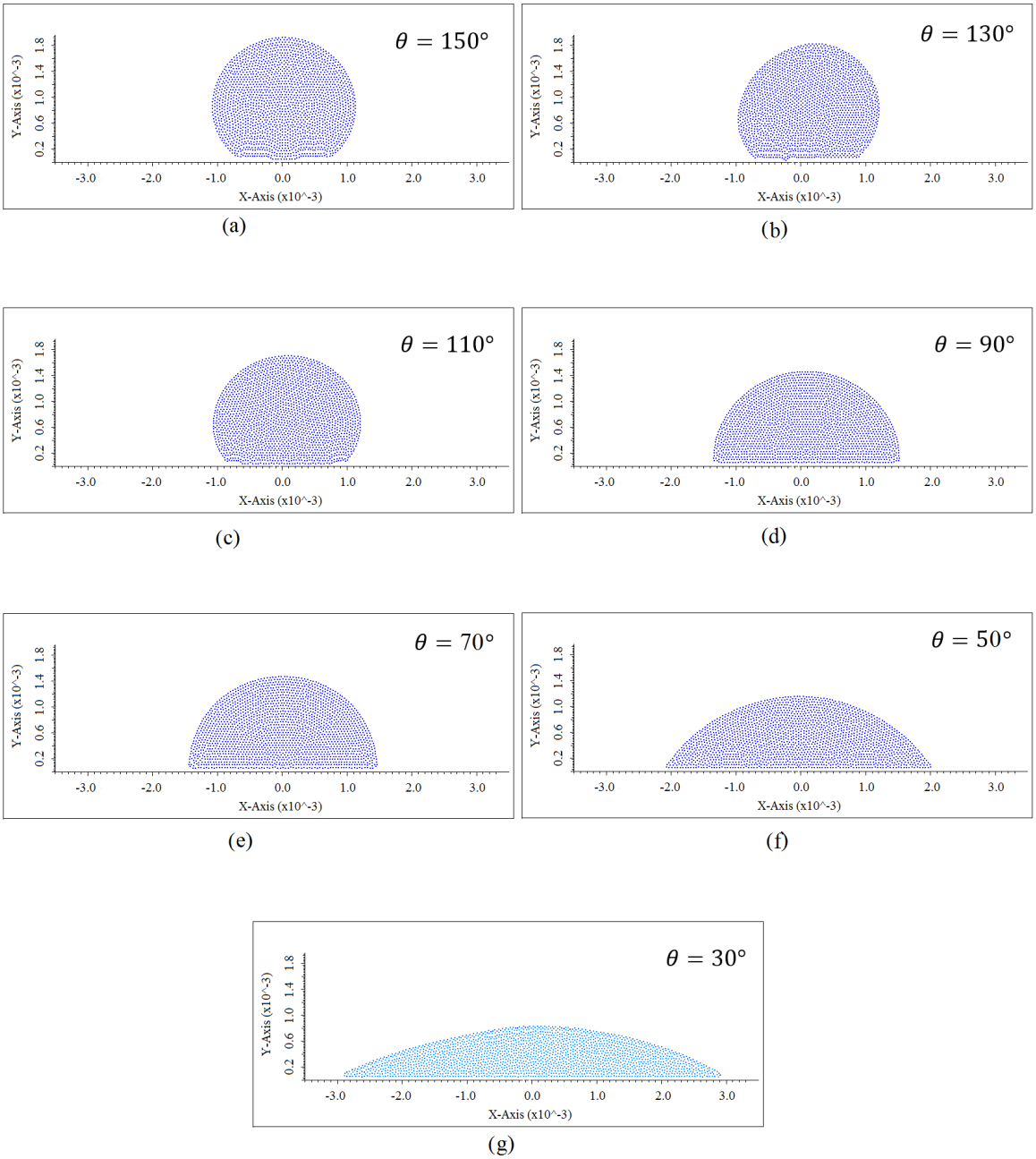
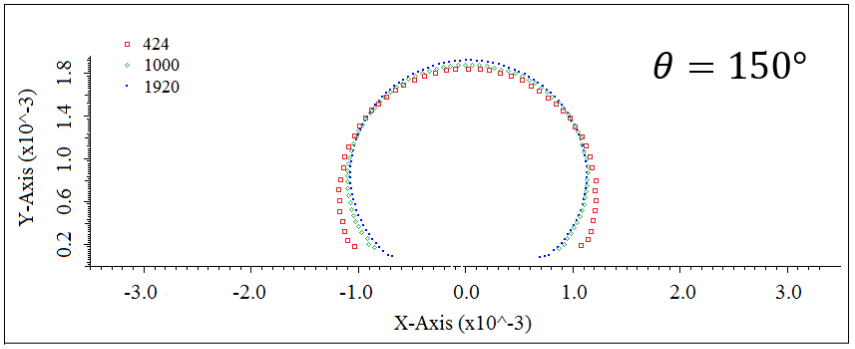
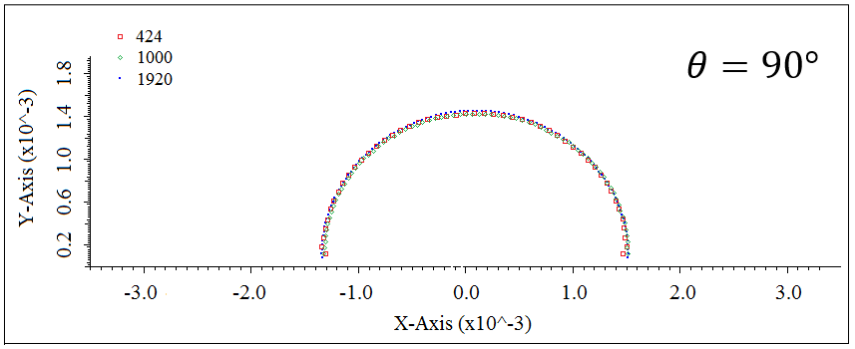


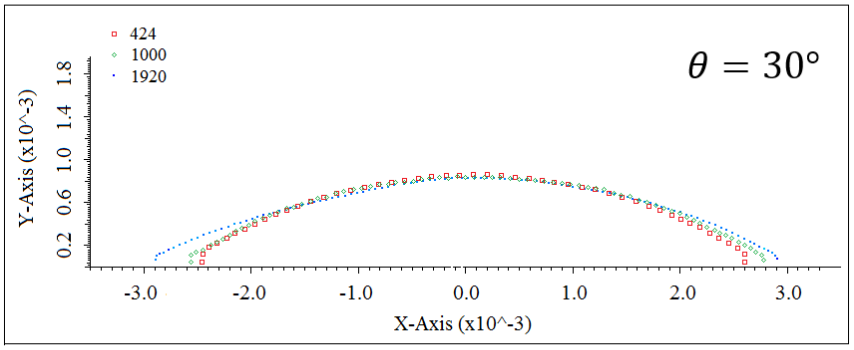
FIGURE 6.21: Contact angle dependance on the set contact angles with the fluid particle resolution of 1920 and with $H^* = 3.5dx$. Static contact angles set to a) 150°, b) 130°, c) 110°, d) 90°, e) 70°, f) 50° and g) 30°



(a)

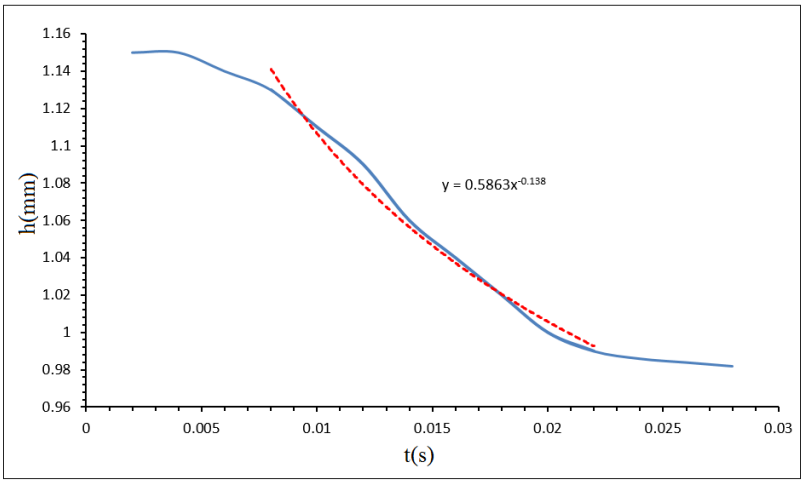


(b)

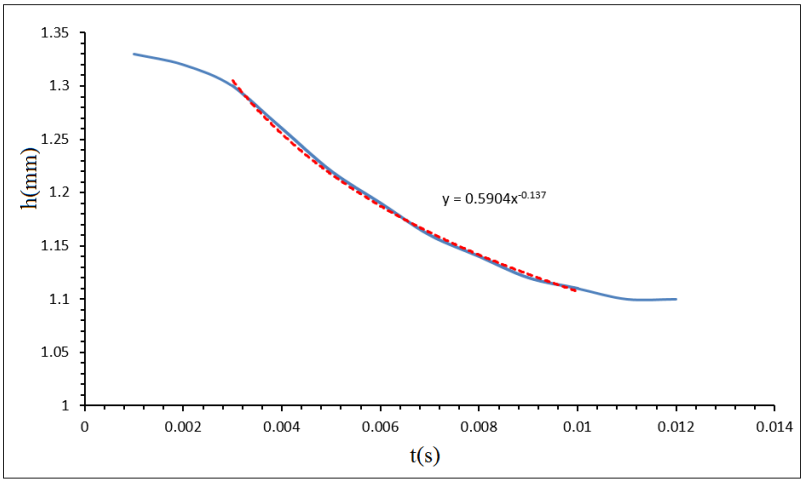


(c)

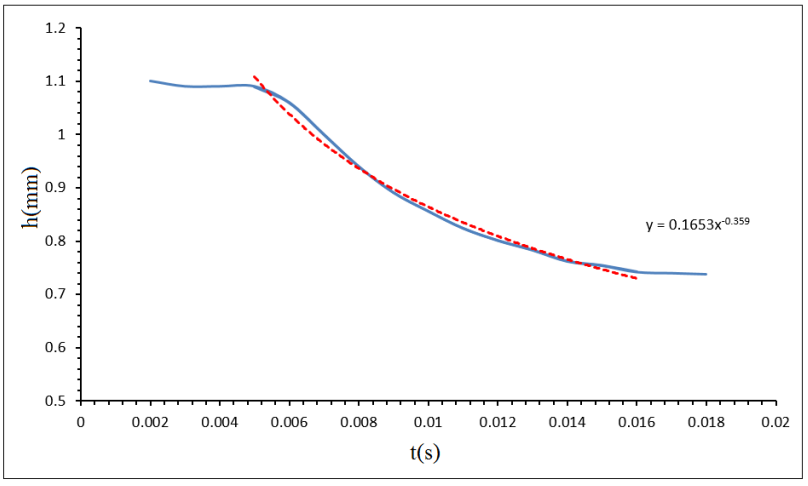
FIGURE 6.22: Comparison of droplet shapes at different particle resolutions for the contact angles of a) 150°, b) 90°, c) 30°.



(a) CLF



(b) Disjoining pressure



(c) IIF

FIGURE 6.23: Evolution of the droplet thickness at the centre of a droplet against time during the spreading process with: a) the CLF method b) the disjoining pressure method c) the IIF method

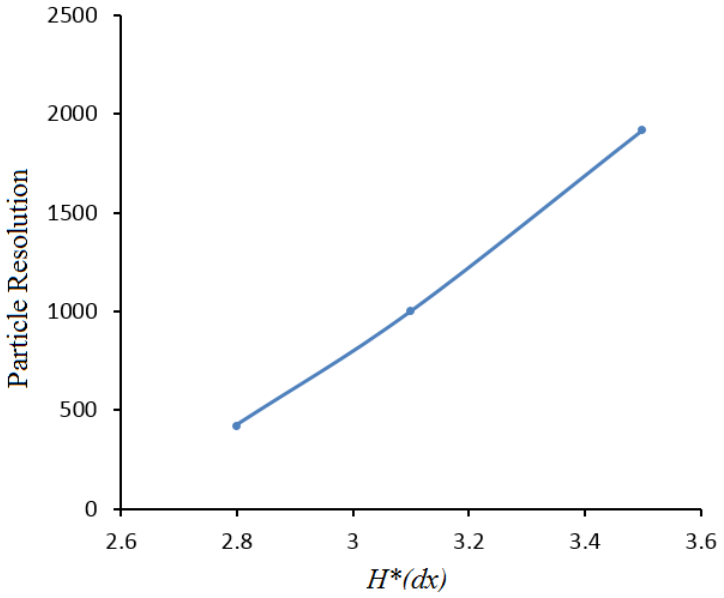


FIGURE 6.24: Constant H^* dependance on the particle resolution.

6.2 Contact Angle Hysteresis

The aim of the present section is to investigate the effect of contact angle hysteresis on droplets. Contact angle hysteresis is the difference in contact angle at both side of a droplet when it is placed on a substrate. Two types of self-agitation, chemical and thermal Marangoni convection, have been reported. Chemical Marangoni is known as a variation of interfacial tension under isothermal conditions and this leads to the spontaneous mechanical movement of a reactive liquid droplet on a substrate. This motion has been predicted theoretically [41, 64] and verified experimentally [10, 32]. Sumino *et al.* [140] conducted several experiments and studied the motion of the oil droplet on the glass substrate such as periodic motion on a straight and narrow glass, as shown in Figure 6.25.

The investigation aims to reproduce comparable results to demonstrate the capability of the SPH methodology using the above experimental findings as test cases to provide a means of verify the potential of simulating dynamic interplay of surface tension forces between the advancing and receding droplet interfaces.

For this study, the IIF method was chosen to control contact angle hysteresis because

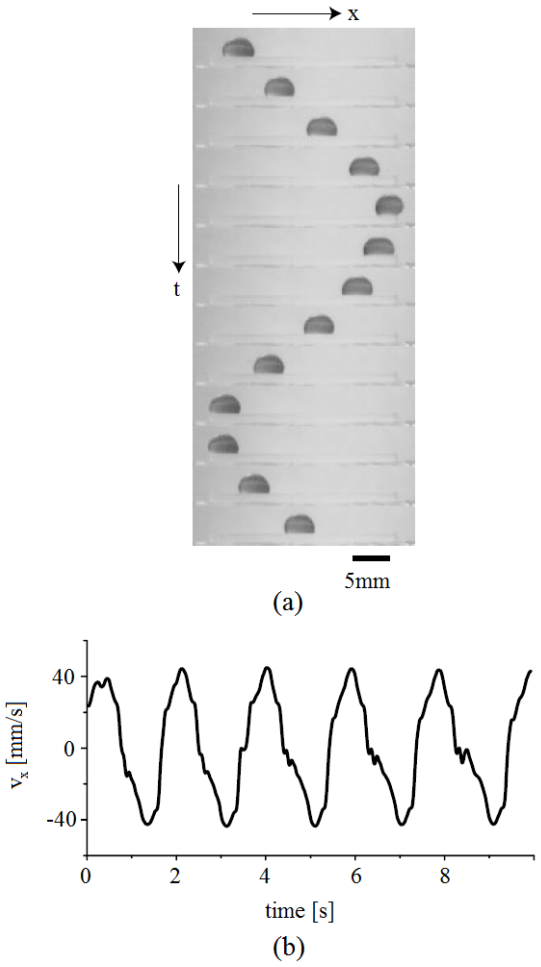


FIGURE 6.25: a) Experiment conducted by Sumino *et al.* [140] that shows the periodic movement of the oil droplet on the glass substrate. b) Change of the oil droplet velocity in x-component in time.

it is simple and is readily implementable into the existing solver than the CLF and disjoining pressure methods. As stated previously in Chapter 5, there is no need to track the surface particles as this consume additional computational resources and affects computational time for long time scale simulations. Also, tracking the surface particles may not be accurate during the evolution of the droplet movement and this may lead to numerical noise resulted from clustering as observed in the CSF and CLF methods. Stable bigger angles greater than 150° up to 180° can be achieved using the IIF method while the CLF and disjoining methods are limited to a maximum of around 140°.

Consider a droplet of radius 2 mm defined on a flat homogeneous substrate with the same static equilibrium contact angles on both sides similar to the one defined in

Subsection 6.1.1. The initial square droplet profile on a flat substrate is allowed to evolve until it reaches equilibrium (see Figure 6.26a). Here, s_{ff} is applied throughout the fluid droplet and s_{sf} is applied between the fluid and solid interface adjacent to the substrate as described in Subsection 6.1.1. Contact angle hysteresis on the droplet is performed by prescribing different s_{sf} values on the substrate. Presently, s_{sf} for both advancing and receding strengths are applied equidistance from the centre of the droplet, as shown in Figure 6.26. This asymmetric interaction force between the substrate and fluid on both sides of the droplet will result in a hysteresis at the interface contact angles. As a result, an unbalanced force is produced from the difference in surface tension pressures (see Young's relation from Equation (2.6)) on the both sides of the droplet contact line interface. This propels the droplet in one direction. The smaller contact angle identified in this study as the advancing angle, θ_A, while the larger one is the receding angle, θ_R.

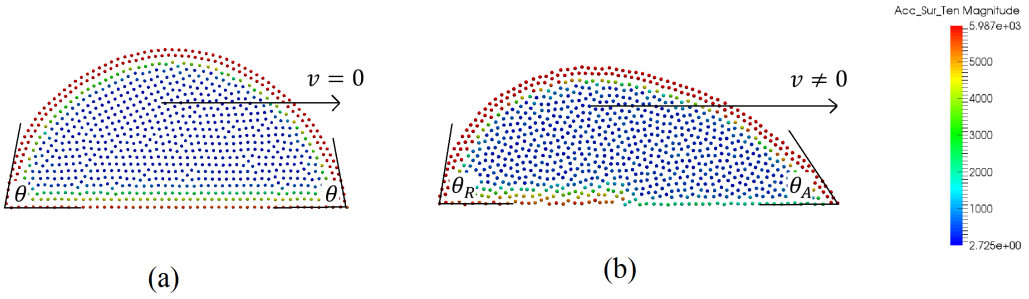


FIGURE 6.26: Schematic diagram of the moving droplet on a substrate due to the difference between the advancing and receding angles ($\theta_R > \theta_A$).

The receding angle is set to $\theta_R = 130°$ while the advancing angle, θ_A, is varied from 130° to 30°. Figure 6.27 shows the rate of change of velocity with time of different receding angles with their respective range of advancing angles. It is observed that the droplet started to move when the difference between the receding and advancing angles becomes greater than $\theta_R - \theta_A = 20°$, as shown in Figure 6.27a. However, in the experiment performed by Chaudhury and Whitesides [32] showed that water droplet begins to move/propelled when the contact angle hysteresis is approximately 10°. In comparison, in numerical model results showed that the droplet achieves self propulsion when $\Delta\theta > 20°$. The driving force that makes the droplet to move becomes greater than the friction force between the droplet and substrate at $\Delta\theta > 20°$. The

droplet velocity increases gradually and becomes constant at time $t = 0.25$ s when the advancing angle approaches 90°. The same trend is observed for the other advancing angles. However, starting from advancing angle of $\theta_A = 50°$, the droplet velocity fluctuates and this is due to the big difference between the receding and advancing angles which causes drag on the droplet.

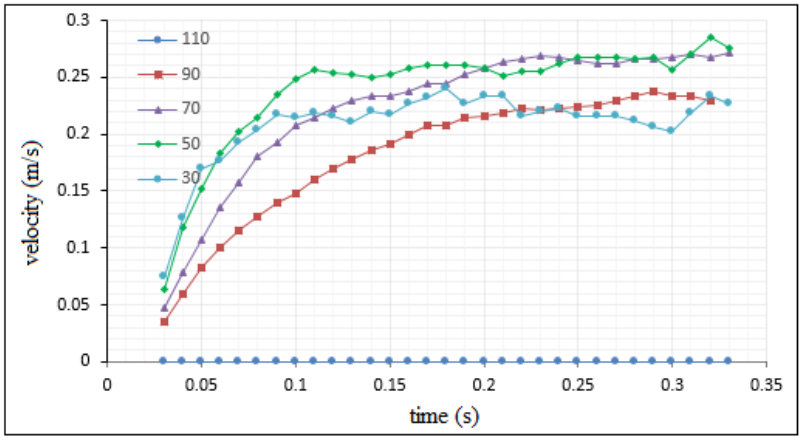
For the receding angle of $\theta_R = 110°$, the advancing angle, θ_A, is varied from 110° to 30°. Overall, the movement that is observed for the advancing angles of $\theta_A = 70°$ is similar to the $\theta_R = 130°$ and $\theta_A = 90°$ result, as shown in Figures 6.27b. The drag phenomena appears starting from the advancing angles of $\theta_A = 50°$ and below.

However, in the case of the receding angle of $\theta_R = 90°$, the advancing angle, θ_A, is varied from 90° to 30°. For this case, the velocity of the droplet increases with time and later remain fairly constant for all advancing angles. This large fluctuations may be due to the large contact area between the droplet and substrate, thus causing greater friction.

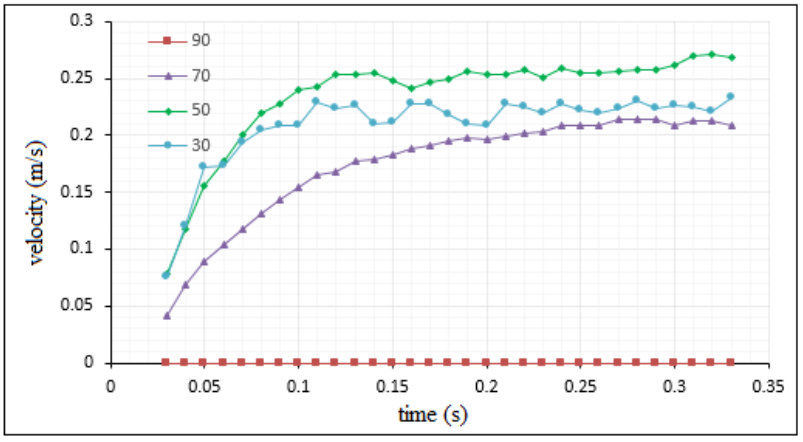
In general, when contact angle hysteresis occurs, and that the droplet begins to move, the droplet accelerates until it reaches a constant velocity. The initial acceleration is greatest when it begins to move and proportional to the size of contact angle hysteresis $\Delta\theta$. However, the trend fades at larger times when the driving force is balanced with shearing force between fluid droplet and wetted substrate. It is observed that contact angles fluctuate at the advancing and receding interfaces and therefore result in velocity oscillation as shown in Figure 6.27 especially for cases when contact angle hysteresis are large. This fluctuation in velocity is particularly obvious when $\Delta\theta > 50°$. In comparison for cases when $\Delta\theta < 50°$ the fluctuations in velocity are negligible and therefore result in smoother acceleration.

Using the above informations and the same size of droplet, periodic motion of the droplet on the substrate is simulated with SPH code. The receding and advancing angles are set at 90° and 50°, respectively. The receding and advancing angles are swapped after each 0.07 s in order to change the direction of the droplet. The test case comparing with the work of Sumino *et al.* [140] was performed to provide a

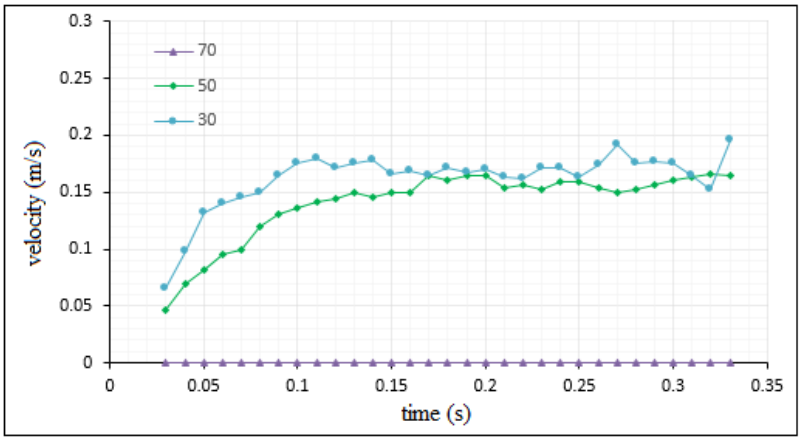
future insight to extending the capability of the SPH methodology and solver. From the simulation the results are qualitatively in good agreement with the experiment performed by Sumino *et al.* [140] as shown in Figure 6.28b. Here, the velocity trend with time is similar to the one observed in the experiment.



(a)



(b)



(c)

FIGURE 6.27: Droplet velocity against time for different advancing angles, θ_A, with receding angles, θ_R, of: a) $\theta_R = 130°$, b) $\theta_R = 110°$, c) $\theta_R = 90°$

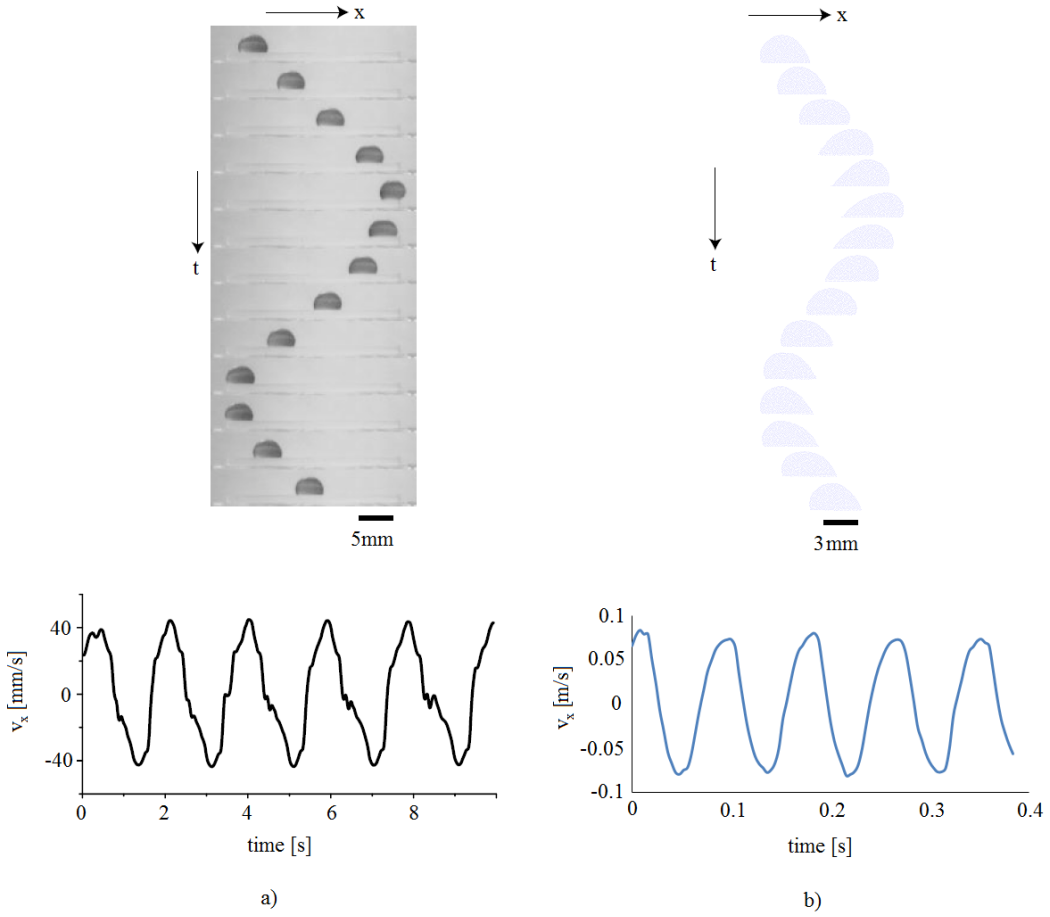


FIGURE 6.28: Qualitative comparison between the experiment and SPH simulation. a) Top: Experiment conducted by Sumino *et al.* [140] that shows the periodic movement of the oil droplet on the glass substrate. Bottom: Change of the oil droplet velocity in x-component in time. b) Periodic motion of the droplet on a substrate simulated in 2D with SPH (top) and its velocity in x-component against time.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, the SPH model was used and developed to accurately predict to simulate surface tension dominated flows, in particular, droplet flows and its evolution with varying contact angle hysteresis. A novel single-phase model is formulated and introduced to account for surface tension effects compared to two-phase model as it requires less computational time since the surrounding air particles are obmitted. The main advantage of using SPH is that it does not require any special treatment to capture the free surface and this makes the method advantageous in handling highly non-linear flows such as wave breaking problems. Unlike mesh based methods, SPH does not need special treatment for wet/dry interfaces. Another advantage of SPH is the straightforward and robustness of the method which does not require the need to generate a mesh for simulating flows with complicated geometry.

The solver is developed from the scratch using the C++ programming language. Four different smoothing kernel functions, such as cubic, quartic, quadratic and fourth order kernel functions are implemented within the numerical solver and can be chosen depending on the nature of the problem being investigated. The developed SPH solver, presented in Chapter 3 solves the mass and momentum conservation equations. All

acceleration terms due to the pressure, viscosity, gravity and surface tension force are computed and included in a convenient way for easy control and accessibility. The developed solver is capable of handling 2D problems, and can be easily extended to model 3D problems with Verlet scheme as a time integration. The time step is chosen according to the Courant-Fredrich-Levy (CFL) condition to ensure that the maximum rate of propagation of parameters do not exceeding the physical rate. A Kd-tree method is used to find the nearest neighbour particles during simulations because it is readily adaptable to various boundary conditions and their requirement.

The accuracy of the developed numerical solver in Chapter 4 via a series of known test cases and to measure its performance. The acceleration due to the pressure term in the momentum equation for the case of an inviscid system is tested using the particle movement inside a box test case, and results are compared and verified with analytical solutions. The result showed that the particle bounce returns to its original height after colliding with the substrate, therefore verifying that system is conserved. The pair of Couette and Poiseuille flow problems are performed to verify viscosity effects with results compared against analytical solutions. The steady state results are in good agreement with maximum error of 0.5% for both problems thus verifying the non-slip condition without the need of special treatment on the boundary layers. The lid driven cavity test case is solved using three different resolutions (50x50, 100x100 and 200x200) for three different Reynolds numbers (Re=100, Re=1000 and Re=10000), showing good result accuracy that approaches traditional mesh based methods with increasing resolutions. The final test case investigates a dam break problem to study the evolution changes of the free surface. The Wendland kernel was chosen due to its soft repulsive property since for problems with free surface it improves the particle distribution and make the surface smoother thereby eliminating the need to use of any additional artificial force to preserve particle ordering. The results obtained from our solver agrees well with experimental results.

In Chapter 5, the developed solver is extended to simulate the formation of droplets in vacuum using two different hybrid surface tension models based on the IIF and CSF approaches. In the IIF model, the predicted droplet formation at equilibrium

state exhibit the presence of unphysical numerical rings and clusterings due to the repulsive and attractive range of the IIF method. Since the repulsive range is less than the attractive range, it causes an unbalanced force to exist, thereby producing the unphysical rings. To solve this problem, two different approaches were used, the first by employing a quadratic kernel function only for momentum due to pressure terms and, the second by using the repulsive force between all particles as an external force in the momentum equation. The repulsive force method was chosen in this thesis due to the more ordered particle distributions. Contrastingly in the CSF model, the above particle ringing problem adjacent to the free-surface does not exist but the methodology requires the surface particles to be tracked and curvature to be computed, thus requiring more computationally time and cost. Unlike the IIF model, the CSF approach require special treatment to solve clusterings problems via introducing the Wendland kernel. A test case using the period for droplet oscillation is employed to examine the accuracy of results obtained from both models and they agreed well with theoretical results.

The effects of how surface tension react on droplets when it is located adjacent to a solid substrate is explored in Chapter 6. Here, the study of droplet spreading with varying contact angles and its evolution is investigated. The IIF, CLF and disjoining pressure approaches were proposed and their performance is studied. In the IIF approach, the contact angle is simply controlled by tuning the interaction strength between the fluid and substrate particles. An advantage of this approach is that there is no requirement to track surface particles which made the methodology simple to deploy. In the CLF approach, the surface particles close to the substrate are detected, tracked and a force is applied in order to control the solid-fluid contact angle while the CSF approach is adopted on the remaining surface particles. In the disjoining pressure approach, high pressure is applied at the solid-liquid interface. The most important advantage of this method is that there no need to track the contact line and disadvantage is that there may appear gap between the solid and substrate and needs to be improved in future. The author believes that disjoining pressure approach is studied here for the first time using SPH. Similar to the CLF approach, the disjoining pressure model requires tracking of surface particles to control the

contact angle. The final problem investigates the effects of contact angle hysteresis. Here, the IIF method is used due to its simplicity, ease of implementation and showed that the methodology is able to handle changes to large and small contact angles. The contact angle hysteresis was studied for different receding angles with varying advancing angles. The change of droplet velocity against time for different contact angle hysteresis shows that the droplet velocity increases initially and then stabilised to a constant velocity.

7.2 Future Work

Three different approaches have been reported in the final chapter of the thesis to study contact angle evolutions. The application of SPH to investigate surface tension dominated flows are still relatively young and various improvements can be done to extend current findings. Among them are:

(i) The contact angle hysteresis with the IIF method can be extended to 3D problems in order to compare with experiments such as droplet running uphill, downhill and climbing steps. Also, the internal circulation of the droplet can be studied with details which is applicable in food and beverage industry.

(ii) The CSF and CLF approaches can be improved in terms of tracking the surface particles to capture higher contact angles. Presently, contact angle of droplets are unstable for angles that higher than 90° because fluid surface particles are located closer to the substrate and tracking particles becomes difficult and this need to be studied in detail to further improve the tracking method. This might also be solved by using higher resolution and/or non-dimensionalising the problem.

(iii) The gap between the droplet and substrate needs to be eliminated in the disjoining pressure approach to enable higher contact angles to be modelled. This proposed disjoining pressure method is limited to static contact angles between

30° and 130°. This can be the result of pressure jump between fluid particles and solid particles. One of the way to solve this problem might be increasing the particle resolution which decreases the particle separation allowing the pressure jump to be smaller.

(iv) The use of parallel computing can be introduced into the solver to reduce the computational time. This allows the problem to be divided into smaller chunks so that the computational load can be distributed across multiple processors on CPUs or GPUs.

Appendix A

Derivation of the Normaliser, Gradient and Laplacian of the poly6 Kernel Function

The efficiency and accuracy of the computation is highly affected by the selection of kernel function in SPH simulation. Basically, the kernel function is a mathematically approximate form of the Dirac δ delta function. The kernel function should decrease to zero as the distance between the interested particle and its neighbour particle increase. The influence region of the kernel function is defined by the smoothing length h, usually $h = 1.3 \cdot dr$ (dr is the initial particle distance).

In this section, a general method to derive normaliser, gradient and laplacian of the poly6 kernel function in both 2D and 3D for the SPH approach is presented, and can be applied in the same way for other kernel functions.

A.1 Normaliser in 2D

Let the poly6 function given as

$$f = (h^2 - r^2)^3 \tag{A.1}$$

Therefore, the poly6 kernel can be written as

$$W = \frac{1}{C} f \tag{A.2}$$

Using the normalisation condition in 2D

$$C = \int_0^h 2\pi r \cdot f \cdot dr \tag{A.3}$$

where $2\pi r$ is the length of circle

$$\begin{aligned}
C &= \int_0^h 2\pi r \cdot (h^2 - r^2)^3 \cdot dr \\
&= \int_0^h 2\pi r \cdot (h^6 - 3h^4 r^2 + 3h^2 r^4 - r^6) \cdot dr \\
&= \int_0^h 2\pi \cdot (h^6 r - 3h^4 r^3 + 3h^2 r^5 - r^7) \cdot dr \\
&= 2\pi \left[\frac{h^6 r^2}{2} - \frac{3h^4 r^4}{4} + \frac{3h^2 r^6}{6} - \frac{r^8}{8} \right] \Bigg|_0^h \\
&= \frac{\pi(4h^8 - 6h^8 + 4h^8 - h^8)}{4} - 0 \\
&= \frac{\pi h^8}{4}
\end{aligned} \tag{A.4}$$

Now, for a 2D case

$$W = \frac{1}{C} f = \frac{4}{\pi h^8} f = \frac{4}{\pi h^8} (h^2 - r^2)^3 \tag{A.5}$$

A.2 Normaliser in 3D

Let the poly6 function given as

$$f = (h^2 - r^2)^3 \tag{A.6}$$

Therefore, the poly6 kernel can be written as

$$W = \frac{1}{C} f \tag{A.7}$$

Using the normalisation condition in 3D

$$C = \int_0^h 4\pi r^2 \cdot f \cdot dr \tag{A.8}$$

where $4\pi r^2$ is the surface area of sphere

$$\begin{aligned} C &= \int_0^h 4\pi r^2 \cdot (h^2 - r^2)^3 \cdot dr \\ &= \int_0^h 4\pi r^2 \cdot (h^6 - 3h^4 r^2 + 3h^2 r^4 - r^6) \cdot dr \\ &= \int_0^h 4\pi \cdot (h^6 r^2 - 3h^4 r^4 + 3h^2 r^6 - r^8) \cdot dr \\ &= 4\pi \left[\frac{h^6 r^3}{3} - \frac{3h^4 r^5}{5} + \frac{3h^2 r^7}{7} - \frac{r^9}{9} \right] \Bigg|_0^h \\ &= \frac{64\pi h^9}{315} \end{aligned} \tag{A.9}$$

Now, for a 3D case

$$W = \frac{1}{C} f = \frac{315}{64\pi h^9} f = \frac{315}{64\pi h^9} (h^2 - r^2)^3 \tag{A.10}$$

A.3 Gradient of the Kernel Function in 2D

Gradient of the poly6 kernel is given as

$$\nabla W = \frac{\partial W}{\partial r} \cdot \frac{\partial r}{\partial x}\mathbf{i} + \frac{\partial W}{\partial r} \cdot \frac{\partial r}{\partial y}\mathbf{j} \tag{A.11}$$

where

$$W = \frac{1}{C} f = \frac{1}{C}(h^2 - r^2)^3 \tag{A.12}$$

with

$$C = \frac{\pi h^8}{4}$$

and

$$\mathbf{r} = x\mathbf{i} + y\mathbf{j}$$

$$r = |\mathbf{r}| = \sqrt{(x^2 + y^2)} = (x^2 + y^2)^{1/2}$$

Derivatives are given by

$$\begin{aligned} \frac{\partial W}{\partial r} &= \frac{\partial \left(\frac{1}{C}(h^2 - r^2)^3\right)}{\partial r} \\ &= \frac{1}{C} \cdot 3(h^2 - r^2)^2 \cdot (-2r) \\ &= -6r(h^2 - r^2)^2 \cdot \frac{1}{C} \end{aligned} \tag{A.13}$$

$$\frac{\partial r}{\partial x} = \frac{\partial((x^2 + y^2)^{1/2})}{\partial x} = \frac{1}{2}(x^2 + y^2)^{1/2-1} \cdot 2x = \frac{x}{(x^2 + y^2)^{1/2}} \tag{A.14}$$

$$\frac{\partial r}{\partial y} = \frac{\partial((x^2 + y^2)^{1/2})}{\partial y} = \frac{1}{2}(x^2 + y^2)^{1/2-1} \cdot 2y = \frac{y}{(x^2 + y^2)^{1/2}} \tag{A.15}$$

By substituting derivatives into Equation (A.11) we obtain

$$\begin{aligned}
\nabla W &= \frac{\partial W}{\partial r}\cdot\frac{\partial r}{\partial x}\mathbf{i} + \frac{\partial W}{\partial r}\cdot\frac{\partial r}{\partial y}\mathbf{j} \\
&= -6r(h^2-r^2)^2\cdot\frac{1}{C}\cdot\frac{x}{(x^2+y^2)^{1/2}}\mathbf{i} + \left(-6r(h^2-r^2)^2\cdot\frac{1}{C}\cdot\frac{y}{(x^2+y^2)^{1/2}}\mathbf{j}\right) \\
&= \frac{1}{C}\cdot\frac{-6r(h^2-r^2)^2}{(x^2+y^2)^{1/2}}(x\mathbf{i}+y\mathbf{j}) \\
&= \frac{1}{C}\cdot\frac{-6r(h^2-r^2)^2}{r}\mathbf{r} \\
&= -6(h^2-r^2)^2\frac{1}{C}\mathbf{r} \qquad \text{(A.16)}
\end{aligned}$$

Finally, gradient of the poly6 kernel function in 2D:

$$\nabla W = -6(h^2-r^2)^2\frac{1}{C}\mathbf{r} = \frac{-24(h^2-r^2)^2}{\pi h^8}\mathbf{r} \qquad \text{(A.17)}$$

A.4 Gradient of the Kernel Function in 3D

Gradient of the poly6 kernel is given as

$$\nabla W = \frac{\partial W}{\partial r} \cdot \frac{\partial r}{\partial x}\mathbf{i} + \frac{\partial W}{\partial r} \cdot \frac{\partial r}{\partial y}\mathbf{j} + \frac{\partial W}{\partial r} \cdot \frac{\partial r}{\partial z}\mathbf{k} \tag{A.18}$$

$$W = \frac{1}{C}f = \frac{1}{C}(h^2 - r^2)^3 \tag{A.19}$$

with

$$C = \frac{64\pi h^9}{315}$$

and

$$\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$$

$$r = |\mathbf{r}| = \sqrt{(x^2 + y^2 + z^2)}$$

Derivatives are given by

$$\begin{aligned} \frac{\partial W}{\partial r} &= \frac{\partial \left(\frac{1}{C}(h^2 - r^2)^3\right)}{\partial r} \\ &= \frac{1}{C} \cdot 3(h^2 - r^2)^2 \cdot (-2r) \\ &= -6r(h^2 - r^2)^2 \cdot \frac{1}{C} \end{aligned} \tag{A.20}$$

$$\frac{\partial r}{\partial x} = \frac{\partial((x^2 + y^2 + z^2)^{1/2})}{\partial x} = \frac{1}{2}(x^2 + y^2 + z^2)^{1/2-1} \cdot 2x = \frac{x}{(x^2 + y^2 + z^2)^{1/2}} \tag{A.21}$$

$$\frac{\partial r}{\partial y} = \frac{\partial((x^2 + y^2 + z^2)^{1/2})}{\partial y} = \frac{1}{2}(x^2 + y^2 + z^2)^{1/2-1} \cdot 2y = \frac{y}{(x^2 + y^2 + z^2)^{1/2}} \tag{A.22}$$

$$\frac{\partial r}{\partial z} = \frac{\partial((x^2 + y^2 + z^2)^{1/2})}{\partial z} = \frac{1}{2}(x^2 + y^2 + z^2)^{1/2-1} \cdot 2z = \frac{z}{(x^2 + y^2 + z^2)^{1/2}} \tag{A.23}$$

By substituting derivatives into Equation (A.18) we obtain

$$\begin{aligned}
\nabla W &= \frac{\partial W}{\partial r}\cdot\frac{\partial r}{\partial x}\mathbf{i} + \frac{\partial W}{\partial r}\cdot\frac{\partial r}{\partial y}\mathbf{j} + \frac{\partial W}{\partial r}\cdot\frac{\partial r}{\partial z}\mathbf{k} \\
&= -6r(h^2-r^2)^2\cdot\frac{1}{C}\cdot\frac{x}{(x^2+y^2+z^2)^{1/2}}\mathbf{i} \\
&+ \left(-6r(h^2-r^2)^2\cdot\frac{1}{C}\cdot\frac{y}{(x^2+y^2+z^2)^{1/2}}\mathbf{j}\right) \\
&+ \left(-6r(h^2-r^2)^2\cdot\frac{1}{C}\cdot\frac{z}{(x^2+y^2+z^2)^{1/2}}\mathbf{k}\right) \\
&= \frac{-6r(h^2-r^2)^2}{(x^2+y^2+z^2)^{1/2}}\cdot\frac{1}{C}(x\mathbf{i}+y\mathbf{j}+z\mathbf{k}) \\
&= \frac{-6r(h^2-r^2)^2}{r}\cdot\frac{1}{C}\mathbf{r} \\
&= -6(h^2-r^2)^2\cdot\frac{1}{C}\mathbf{r} \qquad \text{(A.24)}
\end{aligned}$$

Finally, gradient of the kernel function in 3D:

$$\nabla W = -6(h^2-r^2)^2\frac{1}{C}\mathbf{r} = \frac{-945(h^2-r^2)^2}{32\pi h^9}\mathbf{r} \qquad \text{(A.25)}$$

A.5 Laplacian of the Kernel Function in 2D

Gradient of the poly6 kernel in 2D is given as

$$\nabla W = \frac{-24(h^2 - r^2)^2}{\pi h^8}\mathbf{r} \tag{A.26}$$

Let

$$A = \frac{-24(h^2 - r^2)^2}{\pi h^8} \tag{A.27}$$

Then, Laplacian of the poly6 kernel in 2D can be written as

$$\begin{aligned} \nabla \cdot \nabla W &= \left(\frac{\partial}{\partial x}\mathbf{i} + \frac{\partial}{\partial y}\mathbf{j}\right)(Ax\mathbf{i} + Ay\mathbf{j}) \\ &= \frac{\partial}{\partial x}\mathbf{i} \cdot Ax\mathbf{i} + \frac{\partial}{\partial x}\mathbf{i} \cdot Ay\mathbf{j} + \frac{\partial}{\partial y}\mathbf{j} \cdot Ax\mathbf{i} + \frac{\partial}{\partial y}\mathbf{j} \cdot Ay\mathbf{j} \end{aligned} \tag{A.28}$$

Taking in account that

$$\mathbf{i} \cdot \mathbf{i} = \mathbf{j} \cdot \mathbf{j} = 1$$

$$\mathbf{i} \cdot \mathbf{j} = \mathbf{j} \cdot \mathbf{i} = 0$$

$$\begin{aligned} \nabla \cdot \nabla W &= \frac{\partial}{\partial x}(Ax) + \frac{\partial}{\partial y}(Ay) = \frac{\partial A}{\partial x}x + \frac{\partial x}{\partial x}A + \frac{\partial A}{\partial y}y + \frac{\partial y}{\partial y}A \\ &= \frac{\partial A}{\partial x}x + \frac{\partial A}{\partial y}y + 2A \end{aligned} \tag{A.29}$$

Derivatives are given by

$$\frac{\partial A}{\partial x} = \frac{\left(\frac{-24(h^2-r^2)^2}{\pi h^8}\right)}{\partial x} = \frac{96(h^2 - r^2)x}{\pi h^8} \tag{A.30}$$

$$\frac{\partial A}{\partial y} = \frac{\left(\frac{-24(h^2-r^2)^2}{\pi h^8}\right)}{\partial y} = \frac{96(h^2 - r^2)y}{\pi h^8} \tag{A.31}$$

By substituting derivatives into Equation (A.29) we obtain

$$\nabla \cdot \nabla W = \frac{48}{\pi h^8}(h^2 - r^2)(3r^2 - h^2) \tag{A.32}$$

A.6 Laplacian of the Kernel Function in 3D

Gradient of the poly6 kernel in 3D is given as

$$\nabla W = -6(h^2 - r^2)^2 \frac{1}{C}\mathbf{r} = \frac{-945(h^2 - r^2)^2}{32\pi h^9}\mathbf{r} \tag{A.33}$$

Let

$$A = \frac{-945(h^2 - r^2)^2}{32\pi h^9} \tag{A.34}$$

Then, Laplacian of the poly6 kernel in 3D can be written as

$$\begin{aligned}
\nabla \cdot \nabla W &= \left(\frac{\partial}{\partial x}\mathbf{i} + \frac{\partial}{\partial y}\mathbf{j} + \frac{\partial}{\partial z}\mathbf{k}\right)(Ax\mathbf{i} + Ay\mathbf{j} + Az\mathbf{k}) \\
&= \frac{\partial}{\partial x}\mathbf{i} \cdot Ax\mathbf{i} + \frac{\partial}{\partial x}\mathbf{i} \cdot Ay\mathbf{j} + \frac{\partial}{\partial x}\mathbf{i} \cdot Az\mathbf{k} \\
&+ \frac{\partial}{\partial y}\mathbf{j} \cdot Ax\mathbf{i} + \frac{\partial}{\partial y}\mathbf{j} \cdot Ay\mathbf{j} + \frac{\partial}{\partial y}\mathbf{j} \cdot Az\mathbf{k} \\
&+ \frac{\partial}{\partial z}\mathbf{k} \cdot Ax\mathbf{i} + \frac{\partial}{\partial z}\mathbf{k} \cdot Ay\mathbf{j} + \frac{\partial}{\partial z}\mathbf{k} \cdot Az\mathbf{k}
\end{aligned} \tag{A.35}$$

Taking in account that

$$\mathbf{i} \cdot \mathbf{i} = \mathbf{j} \cdot \mathbf{j} = \mathbf{k} \cdot \mathbf{k} = 1$$

$$\mathbf{i} \cdot \mathbf{j} = \mathbf{j} \cdot \mathbf{k} = \mathbf{k} \cdot \mathbf{i} = 0$$

$$\begin{aligned}\nabla \cdot \nabla W &= \frac{\partial}{\partial x}(Ax) + \frac{\partial}{\partial y}(Ay) + \frac{\partial}{\partial z}(Az) \\ &= \frac{\partial A}{\partial x}x + \frac{\partial x}{\partial x}A + \frac{\partial A}{\partial y}y + \frac{\partial y}{\partial y}A + \frac{\partial A}{\partial z}z + \frac{\partial z}{\partial z}A \\ &= \frac{\partial A}{\partial x}x + \frac{\partial A}{\partial y}y + \frac{\partial A}{\partial z}z + 3A \end{aligned} \tag{A.36}$$

Derivatives are given by

$$\frac{\partial A}{\partial x} = \frac{\left(\frac{-945(h^2-r^2)^2}{32\pi h^9}\right)}{\partial x} = \frac{945(h^2-r^2)x}{8\pi h^9} \tag{A.37}$$

$$\frac{\partial A}{\partial y} = \frac{\left(\frac{-945(h^2-r^2)^2}{32\pi h^9}\right)}{\partial x} = \frac{945(h^2-r^2)y}{8\pi h^9} \tag{A.38}$$

$$\frac{\partial A}{\partial z} = \frac{\left(\frac{-945(h^2-r^2)^2}{32\pi h^9}\right)}{\partial x} = \frac{945(h^2-r^2)z}{8\pi h^9} \tag{A.39}$$

By substituting derivatives into Equation (A.36) we obtain

$$\nabla \cdot \nabla W = \frac{945}{32\pi h^9}(h^2-r^2)(7r^2-3h^2) \tag{A.40}$$

Appendix B

Derivation of the force balance at the contact line zone

The static equilibrium of a drop edge is shown in Figure B.1 below Using the same ideas as Schwartz [135], it is convenient to analyse the fundamental force balance in relation to the equivalent line tension with the disjoining pressure model previously

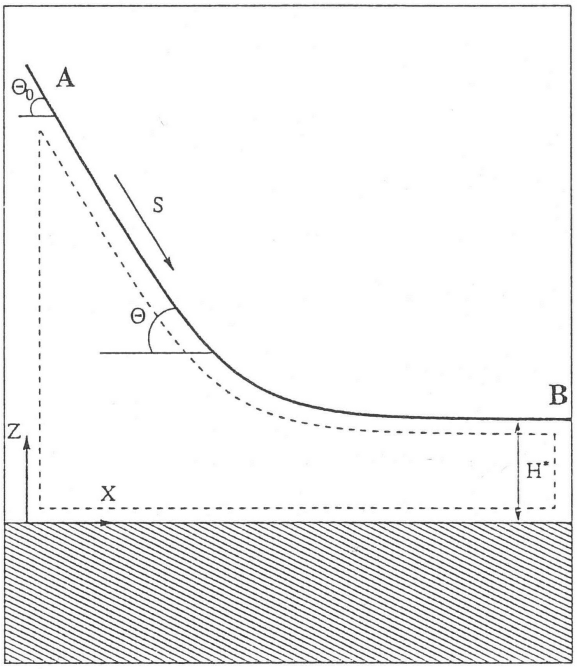


FIGURE B.1: Contact zone of the drop edge at the static equilibrium.

described. Due to fact that the description is limited to the drop edge, only the two-dimensional case will be considered. The point labelled A in Figure B.1 is estimated to be far away from the substrate such that its height H is a multiple of H^*, is at infinite and its $\Pi(H)$ is zero at that point. At equilibrium contact angle θ_0 , the inclination at A remain unchanged, thus making pressure jump created by the curvature of the surface to be zero also, thereby making the total pressure at A to be zero. Similarly, at point B near the precursor film, the inclination θ and it rate of change both are zero with $\Pi(H)$ also zero at this point.

Carryout an integral force balance in the X direction on the region cover by the dashed line in Figure B.1, and knowing that the total pressure at the vertical faces A and B is zero, we came out with the following

$$0 = \int_{H^*}^{\infty} P(H)dH = \sigma \int_{H^*}^{\infty} \frac{d\theta}{dS} dH - \int_{H^*}^{\infty} \Pi(H)dH, \tag{B.1}$$

where S is the arc length measured along the free surface. Since $\frac{dH}{dS} = \sin\theta$, the above equation gives

$$0 = \sigma \cos\theta|_0^{\theta_0} - \int_{H^*}^{\infty} \Pi(H)dH, \tag{B.2}$$

or

$$\sigma \cos\theta_0 = \sigma - E_d(\infty), \tag{B.3}$$

where $E_d(H)$ is the local disjoining energy density and $E_d(\infty)$ is equivalent to the so-called spreading coefficient. Equation (B.3) is the disjoining model identical to Young's Equation (2.6).

Using the two-term disjoining model, the constant B from Equation (6.9) will take the place of θ_0 in Equation (B.2) and will yield

$$B = \frac{(n-1)(m-1)}{H^*(m-n)} \sigma(1 - \cos\theta_0), \tag{B.4}$$

Appendix C

List of the C++ source files

```
#ifndef sphx_common_h
#define sphx_common_h

#include "vector.h"

// Specify dimensions of the problem
typedef c_vector_2d< double > c_vector_xd;

// These constants do not usually require any tweaking
// Do so with care!
#define TOL_NEWTON 1.0e-3
#define VARIABLE_H_ETA 1.3
#define TAITS_GAMMA 7.0
#define TAITS_ETA 0.01
#define GRAVITY 9.81
#define IO_PRECISION 12
#define DT_MAX 100.0
#define rho_ref 1000.0

#define InitialSep (0.003/60.0)

#define m_u 0.001
#define LidSpeed 1.0
#define speedsound 6.0
#define f_cfl 0.04
#define f_a 0.04
#define f_mu 0.04

#define d_x 1.2

#endif
```

```
#include "parameter.h"
#include "common.h"

c_parameter::c_parameter():kernel(""),generator_case("")
{

}

void c_parameter::init()
{
    // Read from file
    kernel = "quintic";             // cubic, quartic, quintic
    generator_case = "test5_3d";
    density = rho_ref;
    density1 = 1.0;
    x_lb = 0.0;
    x_ub = 1.2;
    y_lb = 0.0;
    y_ub = 1.2;
    z_lb = 0.0;
    z_ub = 1.2;
    z_max = z_ub;           // Maximum height of the fluid will travel to define local
sound of sound
    t_begin = 0.0;
    t_end = 0.06;
    dt = 1.0e-6;
    output_filename = "out";
    output_filename_precision = 6;
    write_interval = 0.001;
}
```

```
#include "generator.h"

void generator( std::vector< c_particle_properties > &array, const c_parameter &parameter )
{
    bool status = false;
       if ( parameter.generator_case == "test5_3d" )
    {
        generator_test5_3d( array, parameter );
        status = true;
    }

    if ( status == false )
    {
        std::cout << "generator: test not found!\n";
        exit( EXIT_FAILURE );
    }
}

void generator_test5_3d( std::vector< c_particle_properties > &array, const c_parameter &
parameter )
{
    // Fluid particle generator
    int p_num_x = 60;                                   // number of particles per dimension
    int p_num_y = 60;                                   // number of particles per dimension
    array.reserve( p_num_x * p_num_y );                 // reserve memory to avoid reallocation
    double x_lb = parameter.x_lb;
    double y_lb = parameter.y_lb;
    double x_ub = parameter.x_ub;
    double y_ub = parameter.y_ub;
    double l_x = x_ub - x_lb;
    double l_y = y_ub - y_lb;
    double dx = InitialSep;
    double dy = InitialSep;
    std::size_t counter = 0;
    for ( int i = 0; i < p_num_x; ++i )
    {
        for ( int j = 0; j < p_num_y; ++j )
        {
                double pos_x = x_lb + i * dx;
                double pos_y = y_lb + j * dy;
                c_vector_xd pos( pos_x, pos_y );
                c_vector_xd vel( 0.0, 0.0 );
                c_vector_xd acc( 0.0, 0.0 );
                c_vector_xd vel_old( 0.0, 0.0 );
                double rho = parameter.density;
                double rho_prime = 0.0;
                double rho_old = rho;
                double p = 0.0;
                double mu = m_u;
                double sigma = 0.0728;
                double eta = VARIABLE_H_ETA;
                double m = rho_ref * dx * dy ;
                double h = eta * dx;
                int type = 0;

                c_particle_properties prop;
                prop.pos = pos;
                prop.vel = vel;
                prop.vel_old = vel_old;
                prop.vel_prime = acc;
                prop.m = m;                                 // initial mass
                prop.p = p;                                 // initial pressure
                prop.rho = rho;                             // initial density
                prop.rho_old = rho_old;
                prop.rho_prime = rho_prime;
                prop.mu = mu;                               // intial viscosity
                prop.sigma = sigma;
                prop.h = h;                                 // initial smoothing
length
                prop.type = type;
                prop.idx = counter;
```

```
                array.push_back( prop );
                counter++;
            }
        }

        // Solid particle generator

        int p_num_x_flr = 52;                                   // number of particles per
dimension
        int p_num_y_flr = 1;                                    // number of particles per
dimension
        array.reserve( p_num_x_flr * p_num_y_flr );             // reserve memory to avoid
reallocation
        double x_lb_flr = x_lb ;
        double y_lb_flr = y_lb ;
        for ( int i = 0; i < p_num_x_flr; ++i )
        {
            for ( int j = 0; j < p_num_y_flr; ++j )
            {
                double pos_x = x_lb_flr - 0.0 * dx + i * dx;
                double pos_y = y_lb_flr - 2.0 * dy + j * dy;
                {
                c_vector_xd pos( pos_x, pos_y );
                c_vector_xd vel( 0.0, 0.0 );
                c_vector_xd acc( 0.0, 0.0 );
                c_vector_xd vel_old( 0.0, 0.0 );
                double rho = parameter.density;
                double rho_prime = 0.0;
                double rho_old = rho;
                double p = 0.0;
                double mu = m_u;
                double sigma = 0.0728;
                double eta = VARIABLE_H_ETA;
                double m = rho_ref * dx * dy ;
                double h = eta * dx;
                int type = 1;

                c_particle_properties prop;
                prop.pos = pos;
                prop.vel = vel;
                prop.vel_old = vel_old;
                prop.vel_prime = acc;
                prop.m = m;                                     // initial mass
                prop.p = p;                                     // initial pressure
                prop.rho = rho;                                 // initial density
                prop.rho_old = rho_old;                             // initial
density

                prop.rho_prime = rho_prime;
                prop.mu = mu;                                   // intial viscosity
                prop.sigma = sigma;
                prop.h = h;                                     // initial smoothing
length

                prop.type = type;
                prop.idx = counter;
                array.push_back( prop );
                counter++;
                }
            }
        }
    }
```

```
#ifndef sphx_c_kdtree_h
#define sphx_c_kdtree_h

#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>

// ------------------------------------------------------------
// k-d tree template for a single node
// T1: data class (any type)
// T2: std::vector<double, float, int>
template < typename T1, typename T2 >
struct c_kdnode
{
    T1 data;                                    // store the data of this node - T1: data class
    T2 point;                                   // coordinate of the node - T2: vector<T> class
    c_kdnode< T1, T2 > *left;                   // points to left child node
    c_kdnode< T1, T2 > *right;                  // points to right child node

    c_kdnode();
    c_kdnode( const T1 &leaf_data, const T2 &leaf_point );
};

template < typename T1, typename T2 >
c_kdnode< T1, T2 >::c_kdnode():left(0),right(0),data(),point()
{
}

template < typename T1, typename T2 >
c_kdnode< T1, T2 >::c_kdnode( const T1 &leaf_data, const T2 &leaf_point )
{
    data = leaf_data;                           // assigns data to c_kdnode
    point = leaf_point;                         // assigns point to c_kdnode
    left = 0;                                   // set left pointer to NULL
    right = 0;                                  // set right pointer to NULL
}


// ------------------------------------------------------------
// k-d tree template class
// T1: data class (any type)
// T2: std::vector<double, float, int>
template < typename T1, typename T2 >
class c_kdtree
{
private:
    c_kdnode< T1, T2 > *root;                   // store leaf pointer

public:
    c_kdtree();                                 // constructor
    ~c_kdtree();                                // destructor to remove it from memory
    void insert( const T1 &data, const T2 &point, c_kdnode< T1, T2 > *leaf, int depth );
    // insert leaf routine
    void insert( const T1 &data, const T2 &point );   // interface for insert
    const c_kdnode< T1, T2 > *search_node( const T2 &point, const c_kdnode< T1, T2 > *leaf,
 int depth ); // node search routine
    void search_node( T1 &ret_data, const T2 &point );   // interface for node search
    void search_range( std::vector< const c_kdnode< T1, T2 >* > &vec_point, const T2 &point
, const c_kdnode< T1, T2 > *leaf, double range, int depth );      // range search routine
    void search_range( std::vector< T1 > &ret_data, const T2 &point, double range );    // interface for range search
    void destroy( c_kdnode< T1, T2 > *leaf );   // destroys tree
```

```
    void reset();
};

template < typename T1, typename T2 >
c_kdtree< T1, T2 >::c_kdtree():root(0)
{
}

template < typename T1, typename T2 >
c_kdtree< T1, T2 >::~c_kdtree()
{
    destroy( root );
}

template < typename T1, typename T2 >
void c_kdtree< T1, T2 >::insert( const T1 &data, const T2 &point, c_kdnode< T1, T2 > *leaf,
 int depth )
{
    std::size_t axis = depth % ( point.size() );                    // takes the modulus
    if ( point[axis] < leaf->point[axis] )                          // if point[axis] <
than existing leaf point[axis]
    {
        if ( leaf->left )                                           // if leaf exist,
find insertion point
        {
            insert( data, point, leaf->left, depth + 1 );
        }
        else                                                        // if leaf if empty,
populate it
        {
            leaf->left = new c_kdnode< T1, T2 >( data, point );
        }
    }
    else                                                            // if point[axis] >=
than existing leaf point[axis]
    {
        if ( leaf->right )
        {
            insert( data, point, leaf->right, depth + 1 );
        }
        else
        {
            leaf->right = new c_kdnode< T1, T2 >( data, point );
        }
    }
}

template < typename T1, typename T2 >
void c_kdtree< T1, T2 >::insert( const T1 &data, const T2 &point )
{
    if ( root )                                                     // if tree root
exist, insert in leaf branch
    {
        int depth = 0;                                              // set root of tree
to have depth 0
        insert( data, point, root, depth );                         // calls insert routine
    }
    else
    {
        root = new c_kdnode< T1, T2 >( data, point );               // if no root was
set, assign new tree root
    }
}

template < typename T1, typename T2 >
const c_kdnode< T1, T2 > *c_kdtree< T1, T2 >::search_node( const T2 &point, const c_kdnode<
 T1, T2 > *leaf, int depth )
{
    if ( leaf )                                                     // if the c_kdnode is
not empty
    {
        if ( point == leaf->point )                                 // search found
```

```
        {
            return leaf;                                    // return search leaf
node pointer
        }
        std::size_t axis = depth % ( point.size() );        // takes the modulus
        if ( point[axis] < leaf->point[axis] )              // finds the direction
of the point in tree
        {
            return search_node( point, leaf->left, depth + 1 );
        }
        else
        {
            return search_node( point, leaf->right, depth + 1 );
        }
    }
    else                                                    // if no match found,
return zero pointer (NULL)
    {
        return 0;
    }
}

template < typename T1, typename T2 >
void c_kdtree< T1, T2 >::search_node( T1 &ret_data, const T2 &point )
{
    int depth = 0;                                          // start from depth 0
    const c_kdnode< T1, T2 > *ret_leaf = 0;                 // initialise to NULL
    ret_leaf = search_node( point, root, depth );           // search for leaf node
    if ( ret_leaf )                                         // if found, copy
leaf data into ret_data
    {
        ret_data = ret_leaf->data;
    }
    else                                                    // if not found, or
NULL pointer returned
    {
        std::cout << "k-d tree: Node not found!\n";
    }
}

template < typename T1, typename T2 >
void c_kdtree< T1, T2 >::search_range( std::vector< const c_kdnode< T1, T2 >* > &vec_point,
 const T2 &point, const c_kdnode< T1, T2 > *leaf, double range, int depth )
{
    if ( leaf )
    {
        double norm = 0.0;
        for ( std::size_t i = 0, i_end = point.size(); i < i_end; ++i )
        {                                                   // calculate norm
displacement between the 2 points
            double disp = point[i] - leaf->point[i];
            norm = norm + disp * disp;
        }
        if ( norm <= range * range )                        // using squared of
norm to avoid sqrt operation
        {                                                   // if a node is
within range, store leaf c_kdnode
            vec_point.push_back( leaf );                    // don't know how
many NN, therefore use push_back
        }
        std::size_t axis = depth % ( point.size() );        // takes the modulus
        if ( point[axis] - range < leaf->point[axis] )      // home into the
search point range direction
        {
            search_range( vec_point, point, leaf->left, range, depth + 1 );
        }
        if ( point[axis] + range >= leaf->point[axis] )
        {
            search_range( vec_point, point, leaf->right, range, depth + 1 );
        }
    }
}
```

```
template < typename T1, typename T2 >
void c_kdtree< T1, T2 >::search_range( std::vector< T1 > &ret_data, const T2 &point, double
 range )
{
    int depth = 0;
    std::vector< const c_kdnode< T1, T2 >* > vec_point;
    vec_point.reserve( 100 );
    search_range( vec_point, point, root, range, depth );
    if ( vec_point.size() )
    {
        ret_data.reserve( vec_point.size() );
        for ( std::size_t i = 0, i_end = vec_point.size(); i < i_end; ++i )
        {
            ret_data.push_back( vec_point[i]->data );
        }
    }
    else
    {
    }
}

template < typename T1, typename T2 >
void c_kdtree< T1, T2 >::destroy( c_kdnode<T1, T2> *leaf )
{
    if ( leaf )
    {
        destroy( leaf->left );
        destroy( leaf->right );
        delete leaf;
    }
}

template < typename T1, typename T2 >
void c_kdtree< T1, T2 >::reset()
{
    destroy( root );
    root = 0;
}

#endif
```

```
#include "particle.h"

// ----------------------------------------------------------------
// Particle nearest neighbour
c_particle_neighbour::c_particle_neighbour():idx(0),W(0),grad_W(),lap_W(0)
{

}

// ----------------------------------------------------------------
// Particle properties
c_particle_properties::c_particle_properties():pos(),pos_prime(),vel(),vel_prime(),vel_old
(),m(0),p(0),rho(0),rho_prime(0),rho_old(0),mu(0),sigma(0),h(0),type(0),idx(0),nn(),
VELOCITY(), W_sum(0), vel_prime_surface(), vel_prime_vis(), vel_prime_pres(),
vel_prime_grav(), vel_prime_rep_for()
{
}

// ----------------------------------------------------------------
// Particle kernel
c_particle_kernel::c_particle_kernel():func_W(0),func_deriv_W(0)       // set status to
false and pointers to NULL
{

}

void c_particle_kernel::init( const c_parameter &parameter )
{
    if ( parameter.kernel == "quartic" )
    {
        func_W = W_quartic;                                    // assign template
function to pointer function
        func_deriv_W = deriv_W_quartic;
        func_lap_W = lap_W_quartic;
    }
    if ( parameter.kernel == "cubic" )
    {
        func_W = W_cubic;                                      // assign template
function to pointer function
        func_deriv_W = deriv_W_cubic;
        func_lap_W = lap_W_cubic;
        func_deriv_W_spiky = deriv_W_spiky;
    }
    if ( parameter.kernel == "quintic" )
    {
        func_W = W_quintic;                                    // assign template
function to pointer function
        func_deriv_W = deriv_W_quintic;
        func_lap_W = lap_W_quintic;
        func_deriv_W_spiky = deriv_W_spiky;
    }
    else
    {
        std::cout << "particle kernel not valid.\n";
        exit( EXIT_FAILURE );
    }
}

double c_particle_kernel::W( const c_vector_xd &vec, double h )
{
    return func_W( vec, h );
}

c_vector_xd c_particle_kernel::grad_W( const c_vector_xd &vec, double h )
{
    return grad( func_deriv_W, vec, h );
}

double c_particle_kernel::lap_W( const c_vector_xd &vec, double h )
{
    return func_lap_W( vec, h );
```

```
}

double c_particle_kernel::dWdr( const c_vector_xd &vec, double h )
{
    return func_deriv_W( vec, h );
}


c_vector_xd c_particle_kernel::grad_WP( const c_vector_xd &vec, double h )
{
    return grad( func_deriv_W_spiky, vec, h );
}

// ------------------------------------------------------------
// Particle write
c_particle_write::c_particle_write():status(0),t_interval(0)
{

}

void c_particle_write::xyz( std::ostream &outs, std::vector< c_particle_properties > &array
, double t )
{
    // Write output in XYZ format
    if ( outs.good() )
    {
        // Set output precision
        outs.precision( IO_PRECISION );

        // File header
        outs << "#" << std::endl;
        outs << "# xsph Particle output file" << std::endl;
        outs << "#" << std::endl;
        outs << "# format: px py pz vx vy vz ax ay az m nn t p rho mu h idx" << std::endl;
        outs << "#" << std::endl;

        // Particle write
        outs << "size: " << array.size() << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << std::scientific;
            for ( std::size_t ii = 0, ii_end = array[i].pos.array().size(); ii < ii_end; ++
ii )
            {
                outs << array[i].pos.array()[ii] << " ";
            }
            for ( std::size_t ii = 0, ii_end = array[i].vel.array().size(); ii < ii_end; ++
ii )
            {
                outs << array[i].vel.array()[ii] << " ";
            }
            outs << array[i].m << " " << array[i].nn.size() << " " <<array[i].W_sum <<  " "
 << t << " " << array[i].p << " " << array[i].rho << " " << array[i].mu << " " << array[i].h
 << " " << array[i].idx << std::endl;
        }
        // End of file
        outs << "# End of file" << std::endl;
    }
    else
    {
        std::cout << "writer_xyz: stream state flags (badbit, eofbit or failbit) are
set.\n";
        exit( EXIT_FAILURE );
    }
}

void c_particle_write::vtu( std::ostream &outs, std::vector< c_particle_properties > &array
, double t  )
{
    // Write output in VTU format
    if ( outs.good() )
    {
```

```
        // Set output precision
        outs.precision( IO_PRECISION );

        // File header
        outs << "<?xml version=\"1.0\"?>" << std::endl;
        outs << "<VTKFile type=\"UnstructuredGrid\" version=\"0.1\"
byte_order=\"LittleEndian\">" << std::endl;
        outs << "  <UnstructuredGrid>" << std::endl;
        outs << "    <Piece NumberOfPoints=\"" << array.size() << "\" NumberOfCells=\"" <<
array.size() << "\">" << std::endl;

        // Specify active point data
        outs << "      <PointData Scalars=\"Pressure\" Vectors=\"Velocity\">" << std::endl;

        // Write mass data
        outs << "        <DataArray type=\"Float32\" Name=\"Mass\" format=\"ascii\">" <<
std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << "          " << array[i].m << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        // Write nn data
        outs << "        <DataArray type=\"Float32\" Name=\"NN\" format=\"ascii\">" << std
::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs <<"          " << array[i].nn.size() << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        // Write nn data
        outs << "        <DataArray type=\"Float32\" Name=\"Kernel Sum\"
format=\"ascii\">" << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs <<"          " << array[i].W_sum << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        // Write pressure data
        outs << "        <DataArray type=\"Float32\" Name=\"Time\" format=\"ascii\">" <<
std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << "          " << t << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        // Write pressure data
        outs << "        <DataArray type=\"Float32\" Name=\"Pressure\" format=\"ascii\">"
<< std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << "          " << array[i].p << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        // Write density data
        outs << "        <DataArray type=\"Float32\" Name=\"Density\" format=\"ascii\">"
<< std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << "          " << array[i].rho << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        // Write viscosity data
        outs << "        <DataArray type=\"Float32\" Name=\"Viscosity\" format=\"ascii\">"
 << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
```

```
        {
            outs << "           " << array[i].mu << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        // Write smoothing length data
        outs << "        <DataArray type=\"Float32\" Name=\"Smoothing Length\"
format=\"ascii\">" << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << "           " << array[i].h << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        // Write Kernel Summation
        outs << "        <DataArray type=\"Float32\" Name=\"W_sum\" format=\"ascii\">" <<
std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << "           " << array[i].W_sum << std::endl;
        }
        outs << "        </DataArray>" << std::endl;


        // Write index data
        outs << "        <DataArray type=\"Float32\" Name=\"Index\" format=\"ascii\">" <<
std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << "           " << array[i].idx << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        c_vector_xd temp;
        int dim = temp.d();
        // Write velocity data
        outs << "        <DataArray type=\"Float32\" Name=\"Velocity\"
NumberOfComponents=\"3\" format=\"ascii\">" << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {

            outs << "           " << array[i].vel.x() << " " << array[i].vel.y() << " " <<
0.0 << std::endl;

        }
        outs << "        </DataArray>" << std::endl;


        // Write vel_prime_surface_tension data
        outs << "        <DataArray type=\"Float32\" Name=\"Acc_Sur_Ten\"
NumberOfComponents=\"3\" format=\"ascii\">" << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {

            outs << "           " << array[i].vel_prime_surface.x() << " " << array[i].
vel_prime_surface.y() << " " << 0.0 << std::endl;

        }
        outs << "        </DataArray>" << std::endl;


        // Write vel_prime_viscosity data
        outs << "        <DataArray type=\"Float32\" Name=\"Acc_Visc\"
NumberOfComponents=\"3\" format=\"ascii\">" << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {

            outs << "           " << array[i].vel_prime_vis.x() << " " << array[i].
vel_prime_vis.y() << " " << 0.0 << std::endl;

        }
        outs << "        </DataArray>" << std::endl;
```

```
        // Write vel_prime_pressure data
        outs << "        <DataArray type=\"Float32\" Name=\"Acc_Pres\"
NumberOfComponents=\"3\" format=\"ascii\">" << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {

            outs << "          " << array[i].vel_prime_pres.x() << " " << array[i].
vel_prime_pres.y() << " " << 0.0 << std::endl;

        }
        outs << "        </DataArray>" << std::endl;


        // Write vel_prime_gravity data
        outs << "        <DataArray type=\"Float32\" Name=\"Acc_Grav\"
NumberOfComponents=\"3\" format=\"ascii\">" << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {

            outs << "          " << array[i].vel_prime_grav.x() << " " << array[i].
vel_prime_grav.y() << " " << 0.0 << std::endl;

        }
        outs << "        </DataArray>" << std::endl;

        // Write vel_prime_gravity data
        outs << "        <DataArray type=\"Float32\" Name=\"Acc_Rep_For\"
NumberOfComponents=\"3\" format=\"ascii\">" << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {

            outs << "          " << array[i].vel_prime_rep_for.x() << " " << array[i].
vel_prime_rep_for.y() << " " << 0.0 << std::endl;

        }
        outs << "        </DataArray>" << std::endl;


        // Close point data
        outs << "      </PointData>" << std::endl;


        // Write particle data position
        outs << "      <Points>" << std::endl;
        outs << "        <DataArray type=\"Float32\" Name=\"Position\"
NumberOfComponents=\"3\" format=\"ascii\">" << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << "          " << array[i].pos.x() << " " << array[i].pos.y() << " " <<
0.0 << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        outs << "      </Points>" << std::endl;

        // Write cell data - for particles, the cell is a vertex
        outs << "      <Cells>" << std::endl;
        outs << "        <DataArray type=\"Int32\" Name=\"connectivity\"
format=\"ascii\">" << std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << "          " << i << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        outs << "        <DataArray type=\"Int32\" Name=\"offsets\" format=\"ascii\">" <<
std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << "          " << i + 1 << std::endl;
```

```
        }
        outs << "        </DataArray>" << std::endl;

        outs << "        <DataArray type=\"Int32\" Name=\"types\" format=\"ascii\">" <<
std::endl;
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
            outs << "          " << 1 << std::endl;
        }
        outs << "        </DataArray>" << std::endl;

        outs << "      </Cells>" << std::endl;

        // Close header declarations
        outs << "    </Piece>" << std::endl;
        outs << "  </UnstructuredGrid>" << std::endl;
        outs << "</VTKFile>" << std::endl;
    }
    else
    {
        std::cout << "writer_vtu: stream state flags (badbit, eofbit or failbit) are
set.\n";
        exit( EXIT_FAILURE );
    }
}

void c_particle_write::file( const c_parameter &parameter, std::vector<
c_particle_properties > &array, double t , long c )
{
    if ( status )
    {
        // Append output filename with time tag
        std::string outfile( parameter.output_filename );
        int time_decimal = parameter.output_filename_precision;
        std::string vtu_extension( ".vtu" );
        std::stringstream filename_vtu;
        filename_vtu << outfile << "-time-" << std::fixed << c << vtu_extension;
        std::ofstream f_vtu( filename_vtu.str().c_str() );
        vtu( f_vtu, array, t );
    }
}

void c_particle_write::interval( const c_parameter &parameter, double t, double &dt )
{

    // Adjust dt to produce output at specified interval
    status = false;
    if ( ( t + dt ) >= t_interval )
    {
        dt = t_interval - t;
        t_interval = t_interval + parameter.write_interval;
        status = true;
    }
}

// -----------------------------------------------------------------
// Particle array
c_particle::c_particle():array(),kdtree(),kernel(),write()
{

}

void c_particle::init( const c_parameter &parameter )
{
    // Initialise particle parameters
    kernel.init( parameter );
}

void c_particle::populate( const c_parameter &parameter )
{
    // Populate particles
    generator( array, parameter );
```

```
}

void c_particle::build()
{
    // Create and build k-d tree
    kdtree.reset();
    for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
    {   // only insert the array index data - no need to insert the whole particle data
to determine neighbour
        kdtree.insert( array[i].idx, array[i].pos.array() );
    }
}

void c_particle::nn_find( std::size_t i, double h )
{
    // Determine particle nearest neighbour

    double range = 3.0 * h;                    // range is set at 2h
    std::vector< std::size_t > nn_idx_arr;// define NN index array

    kdtree.search_range( nn_idx_arr, array[i].pos.array(), range );
    array[i].nn.resize( nn_idx_arr.size() );

    for ( std::size_t ii = 0, ii_end = nn_idx_arr.size(); ii < ii_end; ++ii )
    {
        array[i].nn[ii].idx = nn_idx_arr[ii];
    }
}

void c_particle::nn_kernel( std::size_t i, double h )
{
    c_vector_xd r;

    double sum = 0.0;

    for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
    {
        std::size_t j = array[i].nn[ii].idx;

        r = array[i].pos - array[j].pos;
        double r_xx = InitialSep;
        double r_yy = 0.0;
        c_vector_xd r_x( r_xx, r_yy );

        array[i].nn[ii].W = kernel.W( r, h );
        array[i].nn[ii].W_x = kernel.W( r_x, h );
        array[i].nn[ii].grad_W = kernel.grad_W( r, h );
        array[i].nn[ii].grad_WP = kernel.grad_WP( r, h );
        array[i].nn[ii].lap_W = kernel.lap_W( r, h );

        sum = sum + array[j].m / array[j].rho * array[i].nn[ii].W;

    }
    array[i].W_sum = sum;
}

void c_particle::eval_h()
{
    // fixed h, non variable smoothing length
    for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
    {
        double h = array[i].h;
        nn_find( i, h );
        nn_kernel( i, h );
    }

}

void c_particle::eval_rho_prime()
{
    for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
    {
```

```
        if(array[i].type == 0)
        {
            double sum = 0.0;
            for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
            {
                std::size_t j = array[i].nn[ii].idx;
                sum = sum + ( array[i].nn[ii].W ) * array[j].m;
            }
             if( sum < rho_ref )
             {
             sum = rho_ref;
             }
            array[i].rho = sum;
        }
    }
}

void c_particle::eval_pos_prime()
{
    for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
    {
        if ( array[i].type == 0)
        {
            c_vector_xd vec_sum;
            vec_sum.zero();
            for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
            {
                std::size_t j = array[i].nn[ii].idx;
                c_vector_xd v_ij = array[j].vel - array[i].vel;
                double rho_ave = 0.5 * ( array[i].rho + array[j].rho );
                vec_sum = vec_sum + v_ij * ( array[i].nn[ii].W * array[j].m / rho_ave );
            }
            double epsilon = 0.5;
            array[i].pos_prime = array[i].vel + vec_sum * epsilon;
        }
    }
}

void c_particle::eval_pos_prime_1()
{
}

void c_particle::eval_p( const c_parameter &parameter )
{
    for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
    {
        double rho = array[i].rho;
        double rho_0 = parameter.density;
        double gamma = TAITS_GAMMA;
        double c_0 = speedsound;
        double b = c_0 * c_0 * rho_0 / gamma;
        array[i].p = b * ( pow( rho / rho_0, gamma ) - 1.0 );
    }
}

void c_particle::vel_prime_pressure( std::size_t i )
{
    c_vector_xd vec_sum;
    vec_sum.zero();
    for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
    {
        std::size_t j = array[i].nn[ii].idx;
        vec_sum = vec_sum + array[i].nn[ii].grad_W * ( array[j].m * ( array[i].p / ( array[i
].rho * array[i].rho ) + array[j].p / ( array[j].rho * array[j].rho ) ) );
    }
    array[i].vel_prime = array[i].vel_prime - vec_sum;
    array[i].vel_prime_pres = - vec_sum;
```

```
}

void c_particle::vel_prime_gravity( std::size_t i )
{
    // Evaluate particle acceleration due to gravity
    c_vector_xd g( 0.0, -GRAVITY );
    array[i].vel_prime = array[i].vel_prime + g;
    array[i].vel_prime_grav = g;
}

void c_particle::vel_prime_viscous( std::size_t i )
{
    // Evaluate particle acceleration due to viscous effects
    /*
     vec_sum.zero();
     for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
     {
     std::size_t j = array[i].nn[ii].idx;
     c_vector_xd r_ij = array[i].pos - array[j].pos;
     c_vector_xd v_ij = array[i].vel - array[j].vel;
     double zeta = 0.1;
     vec_sum = vec_sum + v_ij * ( r_ij.dot( array[i].nn[ii].dWdr_W ) ) * ( array[j].m * ( array[i].mu + array[j].mu ) ) / ( ( array[i].rho * array[j].rho ) * ( r_ij.norm() * r_ij.norm() + zeta * zeta * array[i].h * array[i].h ) );
     }
     array[i].vel_prime = array[i].vel_prime + vec_sum;
     */

    // alternative to above - see LAMMPS, eqn (23)

    c_vector_xd vec_sum;
    vec_sum.zero();
    for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
    {
        std::size_t j = array[i].nn[ii].idx;
        c_vector_xd r_ij = array[i].pos - array[j].pos;
        c_vector_xd v_ij = array[i].vel - array[j].vel;
        double h = array[i].h;
        double r_norm = r_ij.norm();
        if ( r_norm != 0.0 )
        {
            vec_sum = vec_sum + v_ij * ( array[j].m * ( array[i].mu + array[j].mu ) * kernel.dWdr( r_ij, h ) / ( array[i].rho * array[j].rho * r_norm ) );
        }
    }
    array[i].vel_prime = array[i].vel_prime + vec_sum;
    array[i].vel_prime_vis = vec_sum;

    /*
     // using the laplace approach
     vec_sum.zero();
     for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
     {
     std::size_t j = array[i].nn[ii].idx;
     c_vector_xd v_ij = array[i].vel - array[j].vel;
     vec_sum = vec_sum + v_ij * ( array[i].mu * array[j].m / ( array[i].rho * array[j].rho ) * array[i].nn[ii].lap_W );
     }
     array[i].vel_prime = array[i].vel_prime + vec_sum;
     */
}

void c_particle::vel_prime_surface_tension( std::size_t i )
{

    c_vector_xd vec_sum;
    vec_sum.zero();
    for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
    {
        std::size_t j = array[i].nn[ii].idx;

        if(array[j].type == 0)
```

```
        {
            c_vector_xd r_ij = array[i].pos - array[j].pos;
            double h = array[i].h;
            double r_norm = r_ij.norm();
            double S_ij = 0.008;

            if ( r_norm != 0.0 )
            {
                {
                    if(r_norm <= 3.0*h)
                    {
                        vec_sum = vec_sum + ( r_ij / r_norm ) * S_ij * cos( 1.5 * M_PI *
r_norm / ( 3.0 * h ) );
                    }
                    else
                    {
                        vec_sum = vec_sum;
                    }
                }
            }
        }
    }
    array[i].vel_prime = array[i].vel_prime + vec_sum / array[i].m;
    array[i].vel_prime_surface = vec_sum / array[i].m;
}

void c_particle::vel_prime_repulsive_force( std::size_t i )  // Monaghan 2000
{
    // Evaluate particle acceleration due to pressure gradient
    /* c_vector_xd vec_sum;
     vec_sum.zero();
     for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
     {
     std::size_t j = array[i].nn[ii].idx;
     if(array[j].type == 1)
     {
     vec_sum = vec_sum + array[j].nn[ii].grad_W * ( array[j].m * 0.1 * (
std::abs(array[i].p) / ( array[i].rho * array[i].rho ) + std::abs(array[j].p) / (
array[j].rho * array[j].rho ) ) * pow( array[i].nn[ii].W / array[i].nn[ii].W_x, 4) );
     }
     }
     array[i].vel_prime = array[i].vel_prime - vec_sum;
     array[i].vel_prime_rep_for = - vec_sum ;*/


    c_vector_xd vec_sum;
    vec_sum.zero();
    c_vector_xd r_ij;
    for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
    {
        std::size_t j = array[i].nn[ii].idx;

        if(array[j].type == 0)
        {
            r_ij = array[i].pos - array[j].pos;          // calculate distance between NN
            double r_norm = r_ij.norm();
            if ( r_norm != 0.0 )
            {
                if(r_norm <= InitialSep)
                {
                    vec_sum = vec_sum + ( r_ij * 120.0*InitialSep / ( r_norm ) ) * ( pow((
InitialSep/r_norm ), 12.0) - pow((InitialSep/r_norm ), 6.0) );
                }
                else
                {
                    vec_sum = vec_sum;
                }
            }
        }

    }
    array[i].vel_prime = array[i].vel_prime + vec_sum / array[i].m;
```

```
    //array[i].vel_prime_rep_for = vec_sum / array[i].m ;
}


void c_particle::eval_vel_prime()
{
    for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
    {
        if ( array[i].type == 0 )
        {
            array[i].vel_prime.zero();

            // Evaluate particle acceleration due to pressure gradient
            vel_prime_pressure( i );

            // Evaluate particle acceleration due to gravity
            //vel_prime_gravity( i );

            // Evaluate particle acceleration due to viscous effects
            vel_prime_viscous( i );

            // Evaluate particle acceleration due to surface tension
            vel_prime_surface_tension( i );

            // Evaluate particle acceleration due to repulsive force
            vel_prime_repulsive_force( i );


        }
    }
}

void c_particle::update_pre( double dt )
{
}

void c_particle::update_post( const c_parameter &parameter, double dt, long c, double t )
{
    for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
    {
        // Doing the Verlet algorithm
        double rho_old;
        c_vector_xd vel_old;
        //rho_old = array[i].rho;
        vel_old = array[i].vel;

        if ( c % 50 != 0 )
        {

            if ( array[i].type == 0 )
            {
                //array[i].rho = array[i].rho_old + array[i].rho_prime * ( 2.0 * dt );
                array[i].pos = array[i].pos + array[i].vel * dt + array[i].vel_prime * (
0.5 * dt * dt );
                array[i].vel = array[i].vel_old + array[i].vel_prime * ( 2.0 * dt );
            }
            if ( array[i].type == 1 || array[i].type == 2 )
            {
                // array[i].rho = array[i].rho_old;
                array[i].pos = array[i].pos + array[i].vel * dt ;
                array[i].vel = array[i].vel_old ;
            }
        }
        else
        {

            if ( array[i].type == 0 )
            {
               /// array[i].rho = array[i].rho + array[i].rho_prime * ( dt );
                array[i].pos = array[i].pos + array[i].vel * dt + array[i].vel_prime * (
0.5 * dt * dt );
```

```
                array[i].vel = array[i].vel + array[i].vel_prime * ( dt );
            }
            if ( array[i].type == 1 || array[i].type == 2 )
            {
                //array[i].rho = array[i].rho ;
                array[i].pos = array[i].pos + array[i].vel * dt ;
                array[i].vel = array[i].vel ;
            }
        }

        //array[i].rho_old = rho_old;
        array[i].vel_old = vel_old;


        if( t > 0.01 && t < 0.010001)
        {
            double theta = 0.55 * M_PI ;
            double x = sqrt(2.0/sin(theta)) * array[i].pos.x() * sin(0.5 * theta);
            double y = sqrt(2.0/sin(theta)) * array[i].pos.y() * cos(0.5 * theta);
            c_vector_xd pos_zero(x, y);
                array[i].pos = pos_zero ;
        }
    }
}

void c_particle::time_stepping( const c_parameter &parameter, double &dt )
{
    // Determine time step dt for each particle
    dt = DT_MAX;                        // Resets dt to largest possible dt at start of each
timestep
    for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
    {
        // CFL criterion
        double c_0 = speedsound; //sqrt( g * z_max / TAITS_ETA );    // local speed of
sound of fluid, see Becker et al. 2007
        double dt_cfl = f_cfl * array[i].h / c_0;

        // Acceleration criterion
        double dt_a = f_a * sqrt( array[i].h / array[i].vel_prime.norm() );
        if ( std::isnan( dt_a ) )           // when acceleration = 0
        {
            dt_a = DT_MAX;
        }
        // Viscous criterion
        double dt_mu = f_mu * array[i].h * array[i].h * 1000.0 / array[i].mu;

        //std::cout << array[i].h << std::endl;
        // Return the smallest between particle dt and global dt
        dt = min( dt, min(dt_mu, min( dt_cfl, dt_a )) );
    }
    //dt =0.00001;
}

void c_particle::write_interval( const c_parameter &parameter, double t, double &dt )
{
    // Write data to xyz output
    write.interval( parameter, t, dt );
}

void c_particle::write_output( const c_parameter &parameter, double t, long c )
{
    // Write data to xyz output
    write.file( parameter, array, t, c );
}

void c_particle::density_reinit( const c_parameter &parameter, long c )
{
    if ( c % 30 == 0 )
    {
        for ( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
        {
```

```
            //if(array[i].type == 0)
            //{
                double sum1 = 0.0;
                for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
                {
                    std::size_t j = array[i].nn[ii].idx;
                    sum1 = sum1 + array[j].m / array[j].rho * array[i].nn[ii].W;
                }
                double sum2 = 0.0;
                for ( std::size_t ii = 0, ii_end = array[i].nn.size(); ii < ii_end; ++ii )
                {
                    std::size_t j = array[i].nn[ii].idx;
                    double W_tilda = array[i].nn[ii].W / sum1;
                    sum2 = sum2 + array[j].m * W_tilda;
                }
            if(sum2 < rho_ref)
            {
                sum2=rho_ref;
            }
                array[i].rho = sum2;
            //}
        }
    }
}

void c_particle::dataWipe()
{
    c_vector_xd clear_grad(0.0, 0.0);
    for( std::size_t i = 0, i_end = array.size(); i < i_end; ++i )
    {
        array[i].rho_prime=0.0;
        array[i].vel_prime=clear_grad;
        array[i].pos_prime = clear_grad;
        array[i].nn.clear();

    }
}
```

Bibliography

[1] S. Adami, X.Y. Hu, and N.A. Adams. A new surface-tension formulation for multi-phase SPH using a reproducing divergence approximation. *Journal of Computational Physics*, 229:5011–5021, 2010.

[2] S. Adami, X.Y. Hu, and N.A. Adams. A transport-velocity formulation for smoothed particle hydrodynamics. *Journal of Computational Physics*, 241:292–307, 2013.

[3] A. W. Adamson. *Physical Chemistry of Surfaces*. John Wiley and Sons Inc., New York, 4th edition, 1982.

[4] N. Akinci, G. Akinci, and M. Teschner. Versatile surface tension and adhesion for smoothed particle hydrodynamics fluids. *ACM Transactions on Graphics TOG-Proceedings of ACM SIGGRAPH Asia*, 32, 2013.

[5] P. L. Altman and D. S. Dittmer. *Blood and Other Body Fluids*. Federation of American Societies for Experimental Biology, Bethesda, 3rd edition, 1971.

[6] J.D. Anderson. Computational fluid dynamics: the basics with applications. *Mc Graw-Hill*, 1995.

[7] R. E. Apfel, Y. Tian, J. Jankovsky, T. Shi, X. Chen, R. Glynn Holt, E. Trinh, A. Croonquist, K. C. Thornton, A. Sacco, Jr., C. Coleman, F. W Leslie, and D. H. Matthiesen. Free oscillations and surfactant studies of superdeformed drops in microgravity. *Physical Review Letters*, 78(10):1912–1915, 1997.

[8] S.N. Atluri and S.P. Shen. The meshless local Petrov-Galerkin (MLPG) method. *Tech Science Pree, USA*.

[9] S.N. Atluri and T. Zhu. A new meshless local Petrov-Galerkin (MPLG) approach in computational mechanics. *Computational Mechanics*, 22:117–127, 1998.

[10] C. D. Bain and G. M. Whitesides. A study of contact angle of the acid-base behavior of monolayers containing w-mercaptocarboxylic acids adsorbed on gold: An example of reactive spreading. *Langmuir*, 5:1370–1378, 1989.

[11] G. K. Batchelor. *Introduction to fluid dynamics*. Cambridge University Press., 1974.

[12] T. Belytschko, Y. Krongauz, J. Dolbow, and C. Gerlach. On the comleteness of the meshfree particle methods. *International Journal for Numerical Methods in Engineering*, 43(5):785–819, 1998.

[13] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. Meshless methods: an overview and recently developments. *Computer Methods in Applied Mechanics and Engineering*, 139:3–47, 1996.

[14] T. Belytschko, Y.Y. Lu, and L. Gu. Element-free galerkin methods. *International Journal for Numerical Methods in Engineering*, 37:229–256, 1994.

[15] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[16] W. Benz. Applications of Smoothed Particle Hydrodynamics (SPH) to Astrophysical Problems. *Computer Physics Communications*, 48:130–139, 1988.

[17] W. Benz. Smoothed Particle Hydrodynamics: A review. *Numerical Modeling of Stellar Pulsation: Problems and Prospects In Pro- ceedings of Nato Workshop, Les Arcs, France*, page 269, 1989.

[18] W. Benz. Smoothed Particle Hydrodynamics: a review. *Numerical Modelling of Nonlinear Stellar Pulsations: Problems and Prospects Kluwer Acad. Boston Publ.*, pages 269–288, 1990.

[19] W. Benz and E. Asphaug. Explicit 3D continuum fracture modeling with Smoothed Particle Hydrodynamics. *In Pro- ceedings of Twenty- fourth Lunar and Planetary Science Conference*, pages 99–100, 1993.

[20] W. Benz and E. Asphaug. Impact simulations with fracture. *I. Methods and tests. Icarius*, 107:98–116, 1994.

[21] W. Benz and E. Asphaug. Simulations of brittle solids using Smoothed Particle Hydrodynamics. *Computer Physics Communications*, 87:253–265, 1995.

[22] P. Berczik. Modeling the star formation in galaxies using the chemodynamical SPH code. *Astronomy and Astrophysics*, 360:76–84, 2000.

[23] P. Berczik and I. G. Kolesnik. Smoothed Particle Hydrodynamics and its applications to astrophysical problems. *Kinematics and Physics of Celestial Bodies*, 9:1–11, 1993.

[24] P. Berczik and I. G. Kolesnik. Gas dynamical model for the triaxial protogalaxy collapse. *Astronomy and Astrophysical transactions*, 16:163–185, 1998.

[25] J. C. Berg. Wettability. *Surfactant Science Series. Marcel Dekker Inc., New York*, 49, 1993.

[26] J. J. Bikerman. *Physical Surfaces*. Academic Press, New York, 1970.

[27] J. Bonet and S. Kuasegaram. Corrections and stabilization of Smooth Particle Hydrodynamics methods with applications in metal forming simulations. *International Journal for Numerical Methods in Engineering*, 47:1189–1214, 2000.

[28] J. Bonet and T.-S.L. Lok. Variational and momentum preservation aspects of smooth particle hydrodynamic formulations. *Computer Methods in Applied Mechanics and Engineering*, 180:97–115, 1999.

[29] J. U. Brackbill, D. B. Kothe, and C. A. Zemach. Continuum method for modeling surface tension. *J. Comput. Phys.*, 1992.

[30] V. Butty, D. Poulikakos, and J. Giannakouros. Three-dimensional presolidification heat transfer and fluid dynamics in molten microdroplet deposition. *International Journal of Heat and Fluid Flow*, 23:232–241, 2002.

[31] S. Chandra and C. T. Avedisian. On the collision of a droplet with a solid surface. *Proceedings of the Royal Society of London*, 13:432, 1991.

[32] M. K. Chaudhury and G. M. Whitesides. How to make water run uphill. *Science.*, 256:1539, 1992.

[33] J. K. Chen, J. E. Beraun, and T. C. Carney. A corrective Smoothed Particle Method for boundary value problems in heat conduction. *Computer Methods in Applied Mechanics and Engineering*, 46:231–252, 1999.

[34] N. V. Churaev and V. D. Sobolev. Prediction of contact angle on the basis of the frumkin-derjaguin approach. *Advances in Colloid and Interface Science*, 61:1–16, 1995.

[35] G. Coppola, G. Rocco, and L. de Luca. Insights on the impact of a plane drop on a thin liquid film. *Physics of fluids*, 23:022105, 2011.

[36] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100:32–74, 1928.

[37] A.J.C. Crespo, M. Gomez-Gesteira, and R.A. Dalrymple. Boundary conditions generated by dynamic particle in sph methods. *CMC, Tech Science Press*, pages 173–184, 2007.

[38] S. J. Cummins and M. Menon. An sph projection method. *Journal of Computational Physics*, 152:584–607, 1999.

[39] A. K. Das and P. K. Das. Simulation of drop movement over an inclined surface using smoothed particle hydrodynamics. *Langmuir Article*, 19, 2009.

[40] M. R. Davidson. Spreading of an inviscid drop impacting on a liquid film. *Chemical Engineering Science*, 57:3639–3647, 2002.

[41] P. G. de Gennes. Wetting: Statics and dynamics. *Reviews of modern physics*, 57:827–863, 1985.

[42] P. G. de Gennes, F. Brochard-Wyard, and D. Quere. *Capillarity and Wetting Phenomena: Drops, Bubbles, Pearls, Waves*. Springer, 2002.

[43] W. Dehen and H. Aly. Improving convergence in smoothed particle hydrodynamics simulations without pairing instability. *Mon. Not. R. Astron. Soc.*, pages 1–15, 2012.

[44] B. S. Dooley, A. E. Warncke, M. Gharib, and G. Tryggvason. Vortex ring generation due to the coalescence of a water drop at a free surface. *Experiments in Fluids*, 22(5):369–374, 1997.

[45] C.A. Duarte and J.T. Oden. An HP adaptive method using clouds. *Computer Methods in Applied Mechanics and Engineering*, 139(237-262), 1996.

[46] M. Dupeyrat and E. Nakache. 205 - direct conversion of chemical energy into mechanical energy at an oil water interface. *Bioelectrochemistry and Bioenergetics.*, 5(1):134–141, 1978.

[47] R. H. Durisen, R.A. Gingold, and A. P. Boss. Dynamic Fission Instabilities in Rapidly Rotating n=3/2 Polytropes: A Comparison of Results from Finite-difference and Smoothed Particle Hydrodynamics Codes. *The Astronomical Journal*, 305:281–308, 1986.

[48] A. E. Evrad. Beyond N-body: 3D cosmological gas dynamics. *Mon. Not. R. Astr. Soc*, 235:911–934, 1988.

[49] J. Faber and F. A. Rasio. Post Newtonian SPH Calculations of Binary Neutron Stars Coalescence. Methods and First Results. *Physical Review D*, 62:1–23, 2000.

[50] J. A. Faber and J. B. Manor. Post Newtonian SPH Calculations of Binary Neutron Star Coalescence. II. Mass- ratio, equation of state and spin. *Physical Review D*, 63:1–16, 2001.

[51] O. B. Fawehinmi, P. H. Gaskell, P. K. Jimack, N. Kapur, and H. M. Thompson. A combined experimental and computational fluid dynamics analysis of the dynamics of drop formation. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219:933–947, 2005.

[52] A. Ferrari, M. Dumbser, E. F. Toro, and A. Armanini. A new 3d parallel sph scheme for free surface flows. *Computers and Fluids*, 38:1203–1217, 2009.

[53] G. B. Foote. The water drop rebound problem: dynamics of collision. *Journal of the Atmospheric Science*, 32:390–402, 1975.

[54] A. R. Frederic and C. L. James. Smoothed Particle Hydrodynamics calculations of stellar interactions. *Journal Computational and Applied Mathematics*, 109:213–230, 1999.

[55] A. Frohn and N. Roth. *Dynamics of Droplets*. Springer-Verlag Berlin Heidelberg. Germany, 2000.

[56] J. Fukai, Y. Shiiba, T. Yamamoto, O. Miyatake, D. Poulikakos, C. M. Megaridis, and Z. Zhao. Wetting effects on the spreading of a liquid droplet colliding with a flat surface: Experiments and modeling. *Physics of fluids*, 7:236, 1995.

[57] J. Fukai, Z. Zhao, D. Poulikakos, C. M. Megaridis, and O. Miyatake. Modeling of the deformation of a liquid droplet impinging upon a flat surface. *Physics of fluids*, 5:2588, 1993.

[58] D. A. Fulk. *A Numerical Analysis of Smoothed Particle Hydrodynamics*. PhD thesis, School of Engineering Air University, 1994.

[59] F. Gao and A. A. Sonin. Precise deposition of molten microdrops: The physics of digital microfabrication. *Proceedings of the Royal Society of London*, 444:533, 1994.

[60] P. H. Gaskell, P. K. Jimack, M. Sellier, and H. M. Thompson. Efficient and accurate time adaptive multigrid simulations of droplet spreading. *International Journal for Numerical Methods in Fluids*, 45:1161–1186, 2004.

[61] U. Ghia, K.N. Ghia, and C.T. Shin. High-Resolutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982.

[62] R.A. Gingold and J.J. Monaghan. Smoothed Particle Hydrodynamics: theory and application to non-spherical stars. *Mon. Not. R. astr. Soc.*, 181:375–389, 1977.

[63] P. O. Glantz. The surface tension of saliva. *Ondontologisk Revy.*, 21(2):119–127, 1970.

[64] H. P. Greenspan. On the motion of a small viscous droplet that wets a surface. *Journal of Fluid Mechanics.*, 84:125–143, 1978.

[65] M. Greenspan and M. Yurick. Approximate k-d tree search for efficient icp. *Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM 2003)*, pages 442–448, 2003.

[66] Y.T. Gu and G.R. Liu. A boundary point interpolation method (BPIM) using radial function basis. In *First MIT Conference on Computational Fluid and Solid Mechanics*, pages 1590–1592, MIT, June 2001.

[67] Y.T. Gu and G.R. Liu. A local point interpolation method for static and dynamic analysis of thin beams. *Computer Methods in Applied Mechanics and Engineering*, 190:5515–5528, 2001.

[68] Lopez H. and Leonardo Di G. Sigalotti. Oscillation of viscous drops with smoothed particle hydrodynamics. *Physical Review E*, 73:051201, 2009.

[69] A. Habe. Status rep. super computing japan. *Ed. T. Nakamura, M. Nagasawa. National Lab. High Energy Phys.*

[70] N. Hatta, H. Fujimoto, K. Kinoshita, and H. Takuda. Experimental study of deformation mechanism of a water droplet impinging on hot metalic surfaces above the leidenfrost temperature. *Transactions of the ASME: Journal of Fluids Engineering*, 119:692–699, 1997.

[71] N. Hatta, H. Fujimoto, and H. Takuda. Deformation process of a water droplet impinging on a solid surface. *Transactions of the ASME: Journal of Fluids Engineering*, 117:394, 1995.

[72] M. Herant and W. Benz. Hydrodynamical instabilities and mixing in SN 1987A - Two-dimensional simulations of the first 3 months. *The Astronomical Journal*, 370:81–84, 1991.

[73] C. Hirsch. *Numerical Computation of Internal and External Flows*, volume 1. Wiley-Interscience publication, 1988.

[74] M. Huber, F. Keller, W. Sackel, M. Hirschler, P. Kunz, S. M. Hassanizadeh, and U. Nieken. On the physically based modeling of surface tension and moving contact lines with dynamic contact angles on the continuum scale. *Journal of Computational Physics*, 310:459–477, 2016.

[75] J. P. Hunter. Surface tension in smoothed particle hydrodynamics. *Honours thesis. Math. Dep., Monash Univ*, 1992.

[76] T. Iida and R. I. L. Guthrie. *The Physical Properties of Liquid Metals.* Clarendon Press. Oxford., 1988.

[77] G. R. Johnson, R. A. Stryk, and S. R. Beissel. SPH for high velocity impact computations. *Comput. Methods Appl. Mech. Engineering*, 139:347–373, 1996.

[78] A. Khayyer, H. Gotoh, and S. D. Shao. Corrected incompressible sph method for accurate water-surface tracking in breaking waves. *Coastal Engineering*, 55:236–250, 2008.

[79] J. Kordilla, A. M. Tartakovsky, and T. Geyer. A smoothed particle hydrodynamics model for droplet and film flow on smooth and rough fracture surfaces. *Advances in Water Resources.*, 59:1–14, 2013.

[80] S. Koshizuka and Y. Oka. Moving-particle semi-implicit method for fragmentation of compressible fluid. *Nuclear Science Engineering*, 123:421–434, 1996.

[81] E.S. Lee, C. Moulinec, R. Xu, D. Violeau, D. Laurence, and P. Stansby. Comparisons of weakly compressible and truly incompressible algorithms for the sph mesh free particle method. *Journal of Computational Physics*, 227:8417–8436, 2008.

[82] M. D. Lelah and A. Marmur. Spreading kinetics of drops on glass. *Journal of Colloid and Interface Science*, 82:518–525, 1981.

[83] Z. Levin and P. V. Hobbs. Splashing of water drops on solid and wetted surfaces: Hydrodynamics and charge separation. *Philos. Trans. R. Soc. London*, 269(1200):555–585, 1971.

[84] L. D. Libersky and A. G. Petscheck. Smoothed particle hydrodynamics with strength of materials. *In H. Trease, J. Fritts and W. Crowley (ed.): Proceedings of The Next Lagrange Conference*, 395:248–257, 1991.

[85] L. D. Libersky, A. G. Petscheck, T. C. Carney, J. R. Hipp, and F. A. Allahdadi. High strain lagrangian hydrodynamics: a three-dimensional sph code for dynamic material response. *Journal of Computational Physics*, 109:67–75, 1993.

[86] S. J. Lind, R. Xu, P. K. Stansby, and B. D. Rogers. Incompressible smoothed particle hydrodynamics for free-surface flows: A generalised diffusion-based algorithm for stability and validations for impulsive flows and propagating waves. *Journal of Computational Physics*, 231:1499–1523, 2012.

[87] T. Liszka and J. Orkisz. The finite difference method at arbitrary irregular grids and its applications in applied mechanics. *Computers and Structuresl*, 11:83–95, 1980.

[88] G.R. Liu. *Mesh Free Methods: moving beyond the finite element method*. CRC Press, Boca Raton, 2 edition, 2002.

[89] G.R. Liu and Y.T. Gu. A point interpolation method. In *Proceedings of 4th Asia-Pacific Conference on Computational Mechanics*, pages 1009–1014, December 1999.

[90] G.R. Liu and Y.T. Gu. A local point interpolation method for stress analysis of two-dimensional solids. *Structural Engineering and Mechanics*, 11(2):221–236, 2001.

[91] G.R. Liu and Y.T. Gu. A local radial point interpolation method (LR- PIM) for free vibration analyses of 2-D solids. *Journal of Sound and Vibration*, 246(1):29–46, 2001.

[92] G.R. Liu and Y.T. Gu. A truly meshless method based on the strong-weak form. *Advances in meshfree and X-FEM methods*, pages 259–261, 2002.

[93] G.R. Liu and M.B. Liu. *Smoothed Particle Hydrodynamics a meshfree particle method*. World Scientific, 2003.

[94] G.R. Liu and Quek. The finite element method: a partical course. *Butterworth Heinemann*, 2003.

[95] W.K. Liu and Y. Chen. Wavelet and multiple-scale reproducing kernel methods. *International Journal for Numerical Methods in Fluids*, 21:901–931, 1995.

[96] W.K. Liu, Y. Chen, C.T. Chang, and T. Belytschko. Advances in multiple scale kernel particle methods. *Computational Mechanics*, 18(2):73–111, 1996.

[97] L.B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82-12:1013–1024, 1977.

[98] F. Macia, M. Antuono, and A. Colagrossi. Benefits of using a wendland kernel for free-surface flows. *6th International SPHERIC workshop, Hamburg, Germany*, 2011.

[99] A. Mahdavi and N. Talebbeydokhti. A hybrid solid boundary treatment algorithm for smoothed particle hydrodynamics. *Scientia Iranica*, 22:1457–1469, 2015.

[100] T. Mao, D. C. Kuhn, and H. Tran. Spread and rebound of liquid droplets upon impact on flat surfaces. *AIChE Journal*, 43(9):2169–2179, 1997.

[101] P. Meakin and A. Tartakovsky. Modeling of surface tension and contact angles with smoothed particle hydrodynamics. *Phys. Rev*, 2005.

[102] P. Meakin and A. Tartakovsky. A smoothed particle hydrodynamics model for miscible flow in three-dimensional fractures and the two-dimensional Rayleigh-Taylor instability. *J. Comput. Phys.*, 2005.

[103] P. Meakin and A. Tartakovsky. Modeling and Simulation of Pore-Scale Multiphase Fluid Flow and Reactive Transport in Fractured and Porous Media. *Reviews of Geophysics*, 47(3):1–47, 2009.

[104] Z. Mingyu. *Smoothed Particle Hydrodynamics in Materials Processing: Code Development and Applications.* PhD thesis, Mechanical Engineering, Stony Brook University, May 2007.

[105] V. S. Mitlin and N. V. Petviashvili. Nonlinear dynamics of dewetting - kinetically stable structures. *Physics Letters A*, 192:323–326, 1994.

[106] J. L. Moillet. *Waterproofing and Water-repellency.* Elsevier, Amsterdam, 1963.

[107] A. Mokos. *Multi-phase Modelling of Violent Hydrodynamics Using Smoothed Particle Hydrodynamics (SPH) on Graphics Processing Units (GPUs).* PhD thesis, School of Mechanical, Aerospace and Civil Engineering, 2013.

[108] J. J. Monaghan. Modeling the universe. *In Proceedings of the Astronomical Society of Australia*, 18:233–237, 1990.

[109] J. J. Monaghan. Gravity currents and solitary waves. *Physica D*, 98:523–533, 1996.

[110] J. J. Monaghan, R. F. Cas, A. Kos, and M. Hallworth. Gravity currents descending a ramp in a stratified tank. *Journal Fluid Mechanics*, 379:39–70, 1999.

[111] J. J. Monaghan and A. Kocharyan. SPH simulation of multi-phase flow. *Computer Physics Communications*, 87:225–235, 1995.

[112] J. J. Monaghan and A. Kos. Solitary Waves on a Cretan Beach. *J. Waterway, Port, Coastal and Ocean Engineering*, 125:145–154, 1999.

[113] J. J. Monaghan and A. Kos. Scott Russells Wave Generator. *Physics of fluids*, 12:622–630, 2000.

[114] J. J. Monaghan and J. C. Lattanzio. A simulation of the collapse and fragmentation of cooling molecular clouds. *The Astronomical Journal*, 375:177–189, 1991.

[115] J.J. Monaghan. On the problem of penetration in particle methods. *Journal Computational Physics*, 82:1–15, 1989.

[116] J.J. Monaghan. Smoothed Particle Hydrodynamics. *Annu. Rev. Astron. Astrophys.*, 30:543–574, 1992.

[117] J.J. Monaghan. Simulating free surface flows with SPH. *Journal of Computational Physics*, 110:399–406, 1994.

[118] J.J. Monaghan. Heat conduction with discontinuous conductivity. *Applied Mathematics Reports and Preprints, Monash University*, 18, 1995.

[119] J.J. Monaghan and J.C. Lattanzio. A refined particle method for astrophysical problem. *Astronomy and Astrophysics*, 149:135–143, 1985.

[120] J.P. Morris. A study of the stability properties of SPH. *Applied Mathematics Reports and Preprints, Monash University*, 1994.

[121] J.P. Morris. *Analysis of Smoothed Particle Hydrodynamics with Applications*. PhD thesis, Department of Mathematics, Monash University, 1996.

[122] J.P. Morris. Simulating surface tension with smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 33:333–353, 2000.

[123] J.P. Morris, P.J. Fox, and Y. Zhu. Modeling low reynolds number incompressible flows using sph. *Journal of Computational Physics*, 136:214–226, 1997.

[124] S. Mukherjee and J. Abraham. Crown behavior in drop impact on wet walls. *Physics of fluids*, 19:052103, 2007.

[125] M. Muller, D. Charypar, and M. Gross. Particle-based fluids simulation for interactive applications. *Eurographics/SIGGRAPH Symposium on Computer Animation*, 2003.

[126] M. Nagasawa, T. Nakamura, and S. M. Miyama. Three-dimensional hydrodynamical simulations of type II supernova - Mixing and fragmentation of ejecta. *Astronomical Society of Japan*, 40:691–708, 1988.

[127] B. Nayroles, G. Touzot, and P. Villon. Generalizing the finite element methods: diffuse approximation and diffuse elements. *Computational Mechanics*, 10:307–318, 1992.

[128] S. Nugent and H. A. Posch. Liquid drops and surface tension with smoothed particle applied mechanics. *Physical Review E*, 62(4):4968–4975, October 2000.

[129] E. Onate, S. Idelsohn, O.C. Zienkiewicz, and R.L. Taylor. A finite point method in computational mechanics applications to convective transport and fluid flow. *International Journal for Numerical Methods in Engineering*, 39:3839–3866, 1996.

[130] M. Passandideh-Fard, Y. M. Qiao, S. Chandra, and J. Mostaghimi. Capillary effects during droplet impact on a solid surface. *Physics of Fluids*, 8(3):650–659, 1996.

[131] G. J. Phillips and J.J. Monaghan. A numerical method for three-dimensional simulations of collapsing, isothermal, magnetic gas clouds. *Mon. Not. R. Astr. Soc*, 216:883–895, 1985.

[132] P.W. Randles and L.D. Libersky. Smoothed Particle Hydrodynamics: Some recent improvements and applications. *Computer Methods in Applied Mechanics and Engineering*, 139:375–408, 1996.

[133] B.D. Rogers. 3rd benchmark test: 2-D lid-driven caity flow without gravity, spheric community. *URL: http://wiki.manchester.ac.uk/spheric/*, 2006.

[134] P. Schmuki and M. Laso. On the stability of rivulet flow. *Journal Fluid Mechanics*, 215:125–143, 1990.

[135] L. W. Schwartz. Hysteretic effects in droplet motions on heterogeneous substrate: direct numerical simulation. *Langmuir*, 14:3440–3453, 1998.

[136] M. Sellier. *The Numerical Simulation of Thin Film Flow Over Heterogeneous Substrate*. PhD thesis, School of Mechanical Engineering, 2003.

[137] P. R. Shapiro, H. Martel, J. V. Villumsen, and J. Owen. Adaptive Smoothed Particle Hydrodynamics, with Application to Cosmology: Methodology. *The Astronomical Journal*, 103:269–330, 1996.

[138] J. Shin and T. A. McMahon. The tuning of a splash. *Physics of Fluids: A*, 2(8):1312–1317, 1990.

[139] R. F. Stellingwerf and R. E. Peterkin. Smooth particle magnetohydrodynamics. *Technical report, Albuquerque: Mission Res. Corp.*, 1990.

[140] Y. Sumino, N. Magome, T. Hamada, and K. Yoshikawa. Self-running droplet: Emergence of regular motion from nonequilibrium noise. *Physical Review Letters*, 94:068301, 2005.

[141] J. W. Swegle and S. Attaway. On the feasibility of using Smoothed Particle Hydrodynamics for underwater explotion calculation. *Computational Mechanics*, 17:151–168, 1995.

[142] K. Szewca, J. Pozorskia, and J. P. Minierb. Analysis of the incompressibility constraint in the smoothed particle hydrodynamics method. *Physics. flu-dyn with Preprint submitted to Elsevier on the 27th October*, 2011.

[143] T. Takahashi, H. Yui, and T. Sawada. Direct observation of dynamic molecular behavior at a water/nitrobenzene interface in a chemical oscillation system. *The Journal of Physical Chemistry B.*, 106(9):2314–2318, 2002.

[144] L. H. Tanner. The spreading of silicone oil drops on horizontal surafce. *Journal of Physics D: Applied Physics*, 12:1473–1984, 1979.

[145] A. Tartakovsky and P. Meakin. Modeling of Surface Tension and Contact Angles with Smoothed Particle Hydrodynamics. *Physical Review E*, 72(2), 2005.

[146] A. M. Tartakovsky and A. Panchenko. Pairwise force smoothed particle hydrodynamics model for multiphase flow: surface tension and contact line dynamics. *Journal Computational Physics*, 305:1119–1146, 2015.

[147] A.M. Tartakovsky, K. F. Ferris, and P. Meakin. Lagrangian particle model for multiphase flows. *Computer Physics Communications*, 180:1874–1881, 2009.

[148] G. F. Teletzke. *Thin liquid films: molecular theory and hydrodynamic implications.* PhD thesis, University of Minnesota, 1983.

[149] M. F. Trujillo and S. R. Lewis. Thermal boundary layer analysis corresponding to droplet train impingement. *Physics of fluids*, 24:112102, 2012.

[150] L. Verlet. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Physical Review*, 159(1):98–103, 1967.

[151] D. Violeau. *Fluid Mechanics and the SPH Method.* OXFORD University Press, 2012.

[152] J.G. Wang and G.R. Liu. A point interpolation meshless method based on radial basis functions. *Int. J Numer. Meth. Eng.*, 54:1623–1648, 2002.

[153] R. C. Weast, editor. *CRC Handbook of Chemistry and Physics.* CRC press, Boca Raton, 66th edition, 1985.

[154] D. A. Weiss and A. L. Yarin. Single drop impact onto liquid films: neck distortion, jetting, tiny bubble entrainment, and crown formation. *J. Fluid Mech.*, 385:229–254, 1999.

[155] H. Wendland. Piecewise polynomial, positive definite and compacity supported radial functions of minimal degree. *Adv. Comput. Math.*, 4:389–396, 1995.

[156] M.L. Wilkins. *Computer simulation of dynamic phenomena.* Springer-Verlag Berlin Heidelberg, 1999.

[157] A. M. Worthington. On the forms assumed by drops of liquids falling vertically on a horizontal plate. *Proceedings of the Royal Society of London*, 25:261–272, 1876.

[158] A. M. Worthington. A second paper on the forms assumed by drops of liquids falling vertically on a horizontal plate. *Proceedings of the Royal Society of London*, 25:498–503, 1877.

[159] A. M. Worthington. A study of splashes. *Longsman, Green.*, 1908.

[160] G. Yagawa and T. Yamada. Free mesh method: a new meshless finite element method. *Computational Mechanics*, 18:383–286, 1996.

[161] G. Yagawa and T. Yamada. Meshless method on massively parallel processor with application to fracture mechanics. *Key Engineering Materials*, 145-149:201–210, 1998.

[162] A. L. Yarin and D. A. Weiss. Impact of drops on solid surfaces: self-similar capillary waves, and splashing as a new type of kinetics discontinuity. *Journal of Fluid Mechanics*, 283:141–173, 1995.

[163] L. L. Zheng and H. Zhang. An adaptive level set method for moving-boundary problems: application to droplet spreading and solidification. *Numerical Heat Transfer, Part B*, 37:437–454, 2000.

[164] G. Zhou, W. Ge, and J. Li. A revised surface tension model for macro-scale particle methods. *Powder Technology*, 183:21–26, 2008.

[165] Y. Zhu, PJ. Fox, and JP. Morris. A pore-scale numerical model for flow through porous media. *nternational Journal for Numerical and Analytical Methods in Geomechanics*, 23:881–904, 1999.

[166] O. C. Zienkiewicz and R.L. Taylor. *The Finite Element.* Butterworth-Heinemann, 5 edition, 2000.